Supervised and Semi-Supervised Online Boosting Tree for Industrial Machine Vision Application

Fan Wang, Xinyu Xu, Chang Yuan and P. van Beek

5th International Workshop on Knowledge Discovery from Sensor Data (SensorKDD-2011), Held in conjunction with KDD-2011, San Diego, CA, August 2011

Supervised and Semi-Supervised Online Boosting Tree for Industrial Machine Vision Application

Fan Wang[*]
Dept. Electrical Engineering
Stanford University
fanw@stanford.edu

Xinyu Xu
Sharp Laboratories of
America, Inc.
xxu@sharplabs.com

Chang Yuan
Sharp Laboratories of
America, Inc.
cyuan@sharplabs.com

Peter van Beek
Sharp Laboratories of
America, Inc.
pvanbeek@sharplabs.com

ABSTRACT

Machine learning techniques are being used extensively for knowledge discovery and data mining in industrial inspection applications. Traditionally, all of the training samples are required to be presented for training a classifier at a batch mode. However, in most industrial inspection applications, only a small amount of training samples are available initially and larger amount of them become sequentially available during online prediction. In addition, data properties could be changing over time. We propose supervised and semi-supervised online learning with a Boosting Tree (BT) to adapt and evolve the classifier in an online fashion and thus accommodate new information that becomes available sequentially in industrial inspection applications. The supervised online BT can efficiently expand and update existing BTs to add new knowledge without time consuming batch re-training. The semi-supervised BT utilizes co-training to improve classification accuracy by making use of the information in the unlabeled samples and thus reduce the labeling burden of a human operator. We also proposed compact knowledge representation such that data can be fit into limited memory and a fixed computational complexity can be maintained.

Categories and Subject Descriptors

I.4 [**Image Processing and Computer Vision**]: Applications; I.5 [**Pattern Recognition**]: Models; H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

[*]This work was done when Fan Wang was an intern at Sharp Laboratories of America.

Keywords

Online learning, Boosting tree, Defect detection, Defect classification

1. INTRODUCTION

In nowadays industrial inspection and machine vision systems, tremendous amount of data are acquired from the sensor. Intelligent data mining tasks such as detection and classification are then performed by analyzing the sensor data. Classification plays a central role for tasks such as surface inspection [16, 22], autonomous robotics [22], texture classification [20], and visual tracking[11]. Traditionally, the classifier is trained offline with labeled training data and it is usually kept fixed during the whole online production process. However, in industrial machine vision system, the performance of the classifier may deteriorate over time as new data samples becomes continually available where new data property may be different from initial training set. In order to handle the variations introduced by the newly enrolled samples, the classifier has to be re-trained offline in batch mode using both existing and new data to maintain an acceptable level of performance. The process of re-training can be computationally expensive and time consuming, preventing its applicability in industrial machine vision system which typically requires real time computation with limited memory. Online learning [1, 5, 6, 9, 11, 14, 16, 19, 23] provides an efficient alternative to offline re-training by incrementally updating the classifier knowledge upon the arrival of new data and by establishing a decision boundary that adapts to ever-changing data. However, online schemes are either too adaptive, which causes drifting, or biased by a prior model, which hinders incorporation of novel information.

The first contribution of this paper is that we proposed Online Supervised Boosting Tree (OSBT) to adapt and evolve the classifier in an online fashion for industrial machine vision applications. The proposed OSBT prevents drift by introducing an initial target BT model which is learnt prior to the online learning phase. During online learning, this initial target model is then (1) Updated such that redundant or unnecessary information is gradually filtered out, and important prior knowledge is re-emphasized, and (2) Expanded such that novel or orthogonal information is incorporated.

The two complementary updating and expanding operations can ensure that our online classifier is adaptive to new information responsively, and does not drift away from the prior target. The balance between Updating existing BTs and Adding new BTs is achieved by the proposed cascade prediction.

Our online supervised BT algorithm requires labeled data to achieve good predictive performance. However labeling data during online production is not practical because it is difficult, expensive and time consuming. Semi-supervised learning [10, 12] including self-training [13] and online multiple instance learning [2] addresses this problem by using unlabeled data together with labeled data to improve the performance. Co-Training [3] is a popular semi-supervised learning algorithm that has the assumptions that each example is represented by two or more redundantly sufficient sets of views and additionally these views are independent given the class labels. However, these assumptions are not satisfied in many real-world applications including industrial machine vision application.

The second contribution of this paper is that we proposed Online Semi-supervised Co-Training method to address the problem of robust integration of large amount of unlabeled data to boost prediction performance. In our method, two ensembles of Boosting Tree classifiers co-train each other which requires neither redundant nor independent views. This framework is a general framework of online semi-supervised learner that can work with any ensemble learner (Bagging [4], AdaBoost [7] etc.) to build diverse committees. The main contribution of this method is to meet the hard impractical requirement of Co-Training of two or more independent and redundant feature sets by using a set of diverse, complementary classifiers.

In industrial machine vision applications, data is often tremendous; however, resource, in particular, memory, is scarce. As a result, another problem is how to make use of as much as possible available data while retain memory usage at constant level. Existing methods [6, 18] require to store in memory all the training data, eventually leading to memory explosion. Thus, the third contribution of this paper is that we proposed compact representation of the data to reduce memory requirement based on weighted K-means clustering. Using this method, the knowledge representation remains compact and free from redundancy, so that it can fit into the limited memory and the algorithm maintains a fixed computational complexity.

We demonstrate the efficacy of our algorithms on two industrial inspection tasks: defect detection & defect classification for LCD TV panels. For online supervised learning, we compared the performance of Updating BT classifier, Expanding BT classifier, cascade classifier (i.e. combination of Updating and Expanding classifier), with batch training using precision, recall and F1 score as evaluation metrics. Results on hundreds of LCD TV image data show that the cascade classifier achieves good balance between precision and recall, leading to the highest F1 score. Our results also demonstrate that the semi-supervised Co-Training scheme further increases prediction performance of the online supervised BT by 4%.

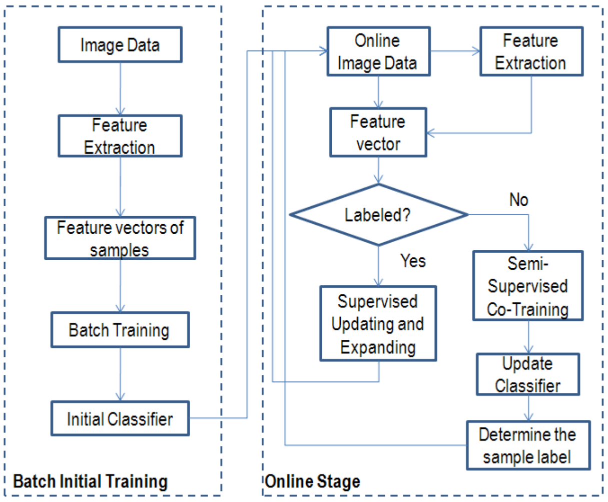


Figure 1: The work flow of the proposed online learning algorithm applied to industrial machine vision system.

The work flow of our online learning algorithm applied to industrial machine vision system is shown in Fig.1. It consists of two phase: offline batch training phase, and online prediction phase. During offline batch training, each image of LCD TV panel is labeled as defective or normal for defect detection application. We then assign defective image a unique defect type label for defect classification application. Next, useful features for defect detection and classification are then extracted and stored as feature vectors along with their labels. We then perform batch training to obtain an initial BT classifier. During online production phase, features are extracted for each incoming image of LCD TV panel, and depending on whether data has label or not, the proposed supervised and semi-supervised classifiers will be used to update the initial BT classifier. The online learning phase will stop if the prediction performance reaches desired level. Operators' feedback on the prediction performance is taken into account to determine when to stop online learning.

Our paper is organized as following: Sec.2 introduces the related work about online supervised and semi-supervised learning. The proposed online Boosting Tree algorithm will be described in Sec.3, followed by its semi-supervised version in Sec.4. A compact representation scheme is proposed to reduce memory requirement in Sec.5. Sec.6 describes the experimental results of automatic defect inspection by the proposed online methods, and finally the paper is concluded by Sec.7.

2. RELATED WORK

2.1 Online (Supervised) Learning

On-line learning is a machine learning paradigm in which an algorithm learns from one instance or sample at a time. Online algorithms have received a lot of attention in the last decade, with several applications ranging from learning complex background and appearance models [1], visual tracking [11], object detection and classification [9, 16], modeling and predicting user preferences [5]. The work in [16] presented two types of online evolving image classifiers: clustering-based classifier and an online fuzzy classifier. The two classifiers are evaluated in industrial surface inspection dealing with CD imprint inspection or metal rotor parts inspection. Online boosting for tracking [11] introduced a scheme where features are selected from a pool of weak classifiers and combined into a strong classifier. In [9] an on-line machine vision system is presented for defect detection in sheet-metal forming processes, however the word "on-line" is related to the classification decisions rather than applying an on-line adaptation procedure for the classifiers.

Several on-line variants of the most popular off-line machine learning algorithms have been proposed in the literature. In [6] the authors propose a solution to the problem of training Support Vector Machines (SVM) with large amount of data. The work of Utgoff et al. [23] introduces incremental decision tree classifiers that can be updated and retrained using new unseen data instances. Several contributions have been proposed to extend the popular AdaBoost algorithm [7] to the online scenario [11, 14, 19].

2.2 Online Semi-supervised Learning

Online learning from partially labeled data should be of a great interest to both data mining and computer vision communities. Various strategies have been proposed to cope with this problem including semi-supervised learning [10, 12], self-training [13], multiple instance learning [2], and co-training [3].

Online semi-supervised boosting [12] is an online version of boosting, where unlabeled examples are labeled greedily using the data adjacency graph. The method learns a binary classifier, where one of the classes tracks the object of interest and the other one models everything else. Online manifold regularization of SVMs [10] is an online learning algorithm for manifold regularization of SVMs. The algorithm learns max-margin classifiers, which are regularized by the data adjacency graph.

In self-training (e.g., [13]) the current classifier evaluates a sample and predicts a label, which is used to update itself. This strategy is applied in most trackers based on on-line learning (e.g., [11]). Even though these samples often characterize correct data that may improve the classifier, this approach is highly prone to drifting. In contrast, by using an oracle (e.g., [17]) drifting can be avoided. An oracle can be considered a classifier, which has a high accuracy while the recall may be low. As a drawback, however, oracles are often too conservative by neglecting orthogonal data, which results in a possible reduction of the information gain.

An alternative way would be to use multiple classifiers that operate on different views and perform co-training (e.g. [3]). In co-training, two independent classifiers, which were trained on a small amount of labeled data, are used to label the unlabeled samples for the other classifier if their own decision is confident. Highly unreliable samples are not used to train the second classifier. In practice, the required independent views are often not available, which violates the theoretical constraints for convergence, and the initial classifiers are too weak to allow for robust learning. Our proposed Semi-Supervised Co-Training with BT overcomes these drawbacks because our initial classifier is a strong ensemble learner whose performance is improved by Boosting, and we use two complementary BT classifiers whose information diversity is sufficient to relax the independent view requirement of classical Co-Training.

3. ONLINE BOOSTING TREE

3.1 Introduction to Boosting Tree

Boosting is a known method based on the use of an ensemble of weak classifiers that is not constrained to specific classifiers. In a Boosting algorithm, a weak classifier is trained with respect to the samples and the associated weights. At each iteration, a weak classifier is added to form a final strong classifier. The weak classifiers are typically weighted by their accuracy. After a weak classifier is added, the samples are re-weighted: the weights of the misclassified samples will be increased, and the samples that are classified correctly will have decreased weights. Weak classifiers that are subsequently added will be trained based on the re-weighted samples, focusing more on the misclassified samples. Please referred to [8] for detailed description.

This section is mainly focused on developing an online Boosting algorithm in a supervised learning framework, which will utilize the ensemble characteristics of Boosting to expand(Sec. 3.2) and update(Sec. 3.3) the old model trained with initial training samples.

If we are using decision tree [21] as the base classifier in the Boosting model, and the feature of each sample is real-valued, all the feature values that have appeared should be stored to update the initial model, which is determined by the characteristics of decision tree. This would be very memory-consuming as the number of samples that must be stored keeps increasing. A compact representation of sample features is utilized for this online framework, which will not lose accuracy while keep the memory usage constant as described in Sec.5.

3.2 Online Boosting Algorithm by Expanding Model

Suppose with the batch of offline training samples, an ensemble classifier of BTs has already been obtained. This initial BT classifier can be expanded as new training samples arrive to incorporate novel information.

With the initial training samples, a model C_0, which is an ensemble of weak classifiers $L_i^0, i = 1, ...M$, is trained. The prediction of C_0 on a testing sample x is made by a weighted combination of the prediction of each weak classifier:

$$C_0(x) = \sum_{j=1}^{M} w_j^0 L_j^0(x) \tag{1}$$

in which $w_j^0, j = 1, ..., M$ are the weights of each classifiers. The label of sample x is given by $\operatorname{sgn}[C_0(x)]$, which is the sign of the output.

In the online stage, the new training samples are collected and stored until there are enough to train another new model. The model trained by the i-th group of samples is denoted

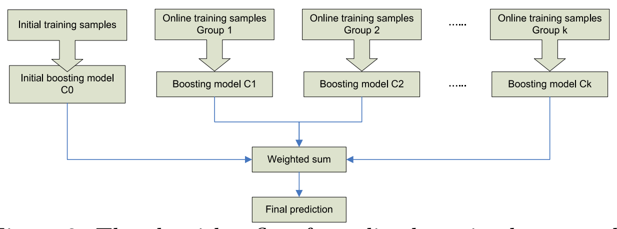

Initial training samples
Online training samples Group 1
Online training samples Group 2
......
Online training samples Group k
Initial boosting model C0
Boosting model C1
Boosting model C2
......
Boosting model Ck
Weighted sum
Final prediction

Figure 2: The algorithm flow for online boosting by expanding BT models.

as C_i, which is also an ensemble of N_i weak classifiers,

$$C_i\left(x\right) = \sum_{i=1}^{N_i} w_j^i L_j^i\left(x\right) \tag{2}$$

The final predicted label C can be calculated as a weighted combination of all the models available up until the current stage, as follows:

$$C = \sum_{i=0}^{K}\left\{\beta_i C_i\left(x\right)\right\} = \sum_{i=0}^{K}\left\{\beta_i\left[\sum_{i=1}^{N_i} w_j^i L_j^i\left(x\right)\right]\right\} \tag{3}$$

where β_i denotes the weight for model C_i.

All the models $C_i, i = 0, ..., K$ are trained with disjointed sets of training samples, so it will be pretty flexible to utilize these models. The weight can be determined by the importance of each group of samples, or by the order that each group of samples arrives, or we can even remove some model if the corresponding group of samples is of little use. The algorithm flow for online boosting by expanding BT models is shown in Fig.2.

This method can also be regarded as a voting mechanism, with each model as an expert, and the final results are determined by them together with different weights.

If the online samples are arriving one by one, we need a buffer to store the samples. As soon as the new samples stored in the buffer are enough to train a new model, we can add the newly trained model and clear the buffer.

3.3 Online Boosting Algorithm By Updating Existing Model

In this part, the initial BT classifier is updated with newly added samples without batch re-training to gradually filter out redundant or unnecessary information and re-emphasize important information. An existing online Boosting algorithm developed in [19] can only update the existing weak classifier and is only suitable for Discrete AdaBoost. In this paper, we proposed an online Boosting algorithm for REAL AdaBoost.

The input of the algorithm is: a group of weak classifier learned so far $L_M^0, i = 1, ..., M$, a new training sample s to arrive (with y_s as its ground truth label), and an incremental learning algorithm which utilizes the new training sample to update the existing BT classifier $BaseUpdate(L_i^0, s)$. The detailed algorithm is described in Algorithm 1.

In the algorithm, W_i is the summation of the weights of all

Algorithm 1 Update Boosting Tree Model

```
Set the new sample's weight as $\lambda_s = 1$;
**for** each weak classifier $L_i^0, i = 1, ..., M$ **do**
    (1) Generate $k$ according to $Poisson\left(\frac{N\lambda_s}{W_i}\right)$;
    (2) Do $k$ times: $L_i^0 = BaseUpdate\left(L_i^0, s\right)$;
    (3) Set
        $$f_i\left(x\right) = \frac{1}{2}\log\frac{p_i\left(x\right)}{1 - p_i\left(x\right)}$$
    in which $p_i\left(x\right) = P\left(y = 1|\, s, L_i^0\right)$;
    (4) Update the new sample's weight
        $$\lambda_s \leftarrow \lambda_s \exp\left(-y_s f_i\left(s\right)\right)$$
    (5) Update the summation of all samples' weights:
        $$W_t = W_t + \lambda_s$$
**end for**
The final prediction of a sample $x$ can be made as
    $$\mathrm{sgn}\left[\sum_{i=1}^{M} f_i\left(x\right)\right]$$
```

samples that have been used until current stage (excluding the weight of new sample s), and N is the number of all past samples excluding the new sample s.

The online Boosting algorithm is similar to its offline version. When the new sample is misclassified by a weak classifier L_i^0, $-y_s f_i\left(s\right)$ would be positive, so the weight λ_s associate with this sample is increased when presented to the next weak classifier; otherwise, the weight λ_s will be decreased.

The basic idea of this online Boosting algorithm is to process the new sample as if it were already included in the initial training set, i.e., also passing it from the first weak classifier to the last, and modifying the sample's weight at each step before passing to the next classifier.

This updating algorithm doesn't require samples to arrive in a group; instead, it can handle the samples one by one, which is different from the algorithm based on expanding models.

3.4 Online Boosting Framework by Cascade Prediction

The previous sections deal with online learning with Updating and Expanding BT models. In this section, we present a prediction scheme that combines the two models to achieve better prediction performance than using any one of them alone. In the following, we denote the expanded model as M_E and denote the updated model as M_U.

If the positive samples and negative samples are unbalanced, the two models will focus on different aspects of the classification aspects. Without losing generality, it is assumed that the number of negative samples is much larger than the number of positive samples, and that finding positive samples is more valuable. Then two different kinds of performance evaluation on positive samples are defined: (1) recall is de-

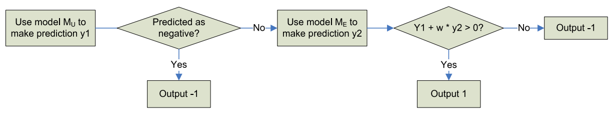

Use model M_U to make prediction y1
Predicted as negative?
No
Yes
Output -1
Use model M_E to make prediction y2
Y1 + w * y2 > 0?
No
Output -1
Yes
Output 1

Figure 3: The illustration of cascade prediction.

fined as the percentage of true positive samples found by the model in all the ground-truth positive samples; (2) precision is defined as the percentage of true positive samples found by the model in all the positive samples predicted by the model.

Experimental results in Sec.6.1 show that M_U will provide results with high positive recall but low precision, and M_E with high precision and low recall. To get both high positive recall and high precision, these two models can be combined together. A basic idea is to make a prediction of a test sample using M_E and M_U separately, and take their summation as the final prediction $\text{sgn}\left[M_E\left(x\right) + M_U\left(x\right)\right]$.

As online stage goes on, M_E will be expanded, so that there will be more and more base weak classifiers in M_E. To make a prediction with M_E, all weak classifiers should be used to make the prediction, which might be time-consuming. Therefore, we proposed cascade prediction which can save a lot of time by avoiding using M_E in unnecessary cases. Cascade prediction is showed in Fig.3. First, we use use M_U to make a prediction; if the sample is predicted as negative, just return -1 as final output; if the sample is predicted as positive, use model M_E to make another prediction, and take the combination of the two predictions as the final output.

As is observed from the experimental results, M_U will provide results with high recall, which means, almost all the positive samples will be included in those samples predicted as positive by M_U. That is to say, if a sample is predicted as negative by M_U, it is most likely that it is a true negative. If it is predicted as positive, we are not so sure whether it is true positive or not; then we combine it with the prediction result by model M_E to make further confirmation. The model M_E will be used only when a sample is predicted as positive in the first stage, which is a small fraction among all samples. Since the number of negative samples is much larger than the number of positive samples, M_U will be enough to make the prediction in many cases, and the time for going through all weak classifiers in M_E will be saved in most cases.

4. ONLINE SEMI-SUPERVISED BOOSTING TREE WITH CO-TRAINING

In this section, we present our online semi-supervised BT method with self-training and co-training to improve predictive performance and reduce labeling burden by making use of large amount of unlabeled samples together with labeled samples. Traditionally, without semi-supervised learning, it is hard to exploit the hidden useful information in the unlabled samples.

The primary unique aspect of this section is a semi-supervised online learning method based on co-training and self-training. Semi-supervised online learning Boosting tree framework based on co-training and self-training. For an application where a limited number of labeled samples are available, but a large number of unlabeled samples is available, it is desirable to use semi-supervised learning to exploit the useful information embedded in unlabeled samples. This means that samples that were not labeled, e.g. by an operator, may be used for training the statistical model of a classifier. This can be highly beneficial in many applications, since there is a cost (in terms of time or other resources) for providing labels for the purpose of training statistical classifiers. Hence, semi-supervised learning reduces cost and/or reduces operator burden.

A Co-Training method [3] that requires neither independent and redundant views nor different learning algorithms is proposed here. In this method, there are two different classifiers M_0 and M_1 trained on labeled samples. Firstly M_0 is used to provide (predict) labels for the unlabeled samples. Then, predicted samples with the most confident labels will be selected, removed from the unlabeled set, and added to the labeled set associated with the predicted labels. M_1 will be re-trained with the expanded labeled set. Then, the role of M_0 and M_1 is exchanged, and the above steps are repeated. In this stage, M_0 will be re-trained based on predictions from the updated M_1. This co-training procedure continues, until some convergence criteria is met.

The most confident samples are selected based on a "maximum difference" rule: if two classifiers M_0 and M_1 have agreed on the sign of the label for a sample, but M_0 predicts larger absolute value (more confident) than M_1, that means M_0 is more confident that M_1 on the label of this sample, then this sample will be defined as the most confident sample and will be provided for updating M_1. It is regarded that this sample can provide the most information to M_1 compared with other samples. Usually, a group of samples satisfying this condition will be selected and provided for updating the other classifier.

In the previous section, two classification models, M_U and M_E, were proposed for online Boosting. Here, we propose that these two models are used in a co-training framework for semi-supervised learning.

With the initial training samples, an initial model M_U is trained. In the following stages, depending on the nature of the coming group of the samples, there are several following operations:

1. If the incoming group of samples is labeled:

   M_U is updated with these samples, and a new Boosting model M_E^i is added to M_E.

2. If the incoming group of samples is unlabeled, and two models M_U and M_E are not empty:

   the co-training procedure is carried out between M_U and the most recently added model M_E^j in M_E.

3. If the coming group of samples is unlabeled and only M_U is ready, M_E is empty:

   A self-training procedure is carried out, which is similar to co-learning, with M_U providing the most confident samples to itself.

The flow of our online semi-supervised Co-Training algorithm is illustrated in Fig.4.

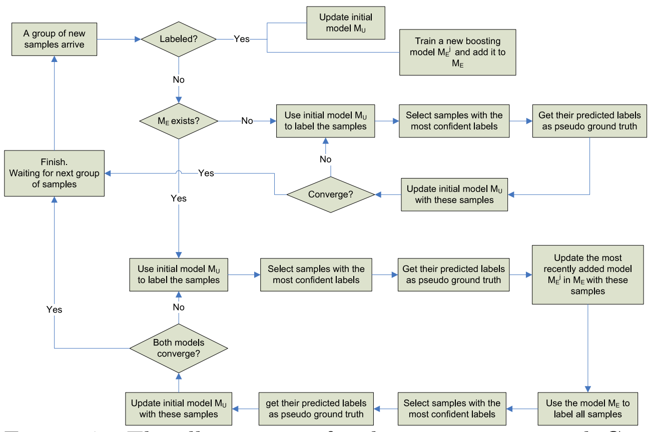

A group of new samples arrive
Labeled?
Yes
Update initial model M_U
Train a new boosting model M_E^j and add it to M_E
No
M_E exists?
No
Use initial model M_U to label the samples
Select samples with the most confident labels
Get their predicted labels as pseudo ground truth
Update initial model M_U with these samples
Converge?
No
Yes
Finish. Waiting for next group of samples
Yes
Use initial model M_U to label the samples
Select samples with the most confident labels
Get their predicted labels as pseudo ground truth
Update the most recently added model M_E^j in M_E with these samples
Use the model M_E to label all samples
Select samples with the most confident labels
get their predicted labels as pseudo ground truth
Update initial model M_U with these samples
Both models converge?
No
Yes

Figure 4: The illustration of online semi-supervised Co-Training algorithm.

If the online samples are arriving one by one instead of group by group, a buffer is needed to store the labeled samples until there are enough to train a new model to be added. There is no need to hold another buffer for unlabeled samples, since updating the existing model and the co-training framework can handle the samples one by one.

In the prediction stage of using the classifier, the updated initial model M_U and each of the extension models M_E^i produce a prediction label. Then these labels are combined by a weighted summation to generate the final predicted label.

5. COMPACT DATA REPRESENTATION

If the base weak classifier used in a Boosting model is the decision tree, then we need to maintain enough information about previous samples in order to update each tree. Specifically, such information is needed to recalculate the purity score of each node. For variables with discrete values, itąŕs easy to obtain this statistical information; however, if the variables are real-valued, to precisely maintain the distribution information, all the previous feature values that have appeared should be stored for future use. This represents a big storage cost.

A compact representation of real-valued sample features is proposed here, based on performing a weighted k-means clustering algorithm on the samples. This method considers the cluster centers as representative samples with weights, and calculates the new centers by taking the sample weights into consideration.

In the above, an algorithm for updating the existing weak classifier $BaseUpdate\left(L_i^0, s\right)$ was mentioned. This algorithm is actually algorithm-specific (specific to the type of baseline classifier). Take the decision tree classifier [21] as an example. If we already have a decision tree trained, and want to update it with a new sample, we need to first pass the sample from the root to corresponding branch according to the criteria at each internal node, and recalculate the "purity" score for each node. If the purity score of a node is not high enough, it should be re-split based on all previous samples and this new sample. Therefore, some information should be maintained about previous samples so that it's possible to recalculate the purity score of each node. For variables with discrete value, it's easy to get this statistical information, just by counting the number of samples with each value; however, if the variable is real-valued, to precisely maintain the distribution information, all the previous feature values that have appeared should be stored for future use.

In online learning, more and more training samples become available over time, so the growing size of the training set makes the training process space-consuming and computationally costly. In our paper, we noted that the information contained in the training set can be represented by the distribution of the training sample, which can be further quantized into a set of discrete representative samples with weights.

In the initial stage, we obtain the representative samples of initial training samples by using the K-means clustering algorithm [15] on positive and negative samples separately. The cluster centers are selected as the representative samples, denoted as $\{\hat{x}_i, i = 1, ..., n\}$, and their weights are taken as the number of samples in the corresponding cluster, denoted as $\{s_i, i = 1, ..., n\}$.

In the online stage, the new training samples are added to the previous representative sample set $\{\hat{x}_i, i = 1, ..., n\}$ with each new training sample assigned a weight of 1. A weighted k-means is performed on this enlarged set as described in Algorithm 2.

Algorithm 2 Weighted K-Means for compact sample representation

```
Randomly select $n$ points as the initial cluster centers $\{center_i, i = 1, ..., n\}$;
Assign each sample $x$ to the nearest cluster center:
    $$i = \arg\min_k \|x - center_k\|_2 \Rightarrow x \in C_i$$
while Error is above some threshold do
    (1) Re-calculate the cluster center by taking the weighted average of all samples in the cluster:
        $$center'_i = \sum_{x_j \in C_i} s_j x_j \Bigg/ \sum_{x_j \in C_i} s_j$$
    and the weight associated with the center is
        $$s'_i = \sum_{x_j \in C_i} s_j$$
    (2) Replace the old centers $\{center_i, i = 1, ..., n\}$ with new centers $\{center'_i, i = 1, ..., n\}$, also replace the associated weights.
    (3) Calculate the cluster error:
        $$error = \sum_{i=1}^{n} \sum_{x_j \in C_i} s_j \|x_j - center_i\|_2^2$$
end while
Return the up-to-date cluster centers $\{center_i, i = 1, ..., n\}$ and the associated weights $\{s_i, i = 1, ..., n\}$ as the compact representation of all samples
```

Using this method, the statistical distribution information contained in the entire sample set is dynamically maintained, while the memory footprint and the computational load is controlled as constant, proportional to the number of centers used in the K-means clustering algorithm. This method can be regarded as a quantization in least-square sense.

Although the compact represented is described here for the purpose of updating decision trees, this representation is also useful for other types of classifiers, as long as the classifier will be influenced by the data density, and as long as it is important to maintain the distribution information for future use.

6. EXPERIMENTAL RESULTS

We demonstrate the efficacy of our algorithms on two industrial inspection tasks: defect detection and defect classification for LCD TV panels.

6.1 Online Boosting Tree for Defect Detection

In this application, the training and testing data is obtained from LCD panel production line. The goal is to detect defective pixels from images of LCD TV panel. Each image approximately has 10,000 to 30,000 pixels, and defective region has been labeled with a binary mask image. In this application, usually the number of positive (defective) samples is much more than nagative (normal) samples. The samples in each image are used as a group to train the Boosting Tree. In total there are 117 images, there are divided into 3 sets: initial offline training set (25 images), online training set (25 images) becoming available one by one at 25 online training stages, and testing set (67 images) which are unseen to the classifier.

We used recall, precision and F1 score as our evaluation metric. The performance of the 3 metrics is shown in Fig.5. These performance scores are reported on the whole data set including all pixels in 117 images. F1 score is the harmonic mean of precision and recall, while precision and recall have been defined in Sec.3.4.

Each curve contains 26 points, where the first point at stage 0 represents the performance of the model trained with offline training samples, and the other 25 points stand for performance of the 25 online stages. The blue line represents performance obtained by expanding BT model, magenta line represents the performance obtained by updating the existing BT model, and the red line corresponds to the performance of cascade combination of Expanding and Updating BT model. The solid green line stands for the performance of BT model whose size is the same as the model expanded after 25 stages. The performance of solid green line is the best we can achieve with all available training samples (initial offline plus online), but it is impractical in industrial inspection application. The dashed green line shows the performance of the BT model whose size is the same as the initial model size. The model size is defined as the number of weak classifiers (Decision Tree) used in the BT.

Our first observation is that performance is increasing as more online samples become available. Cascade model combines the benefits of expanding model and updating existing model, leading to highest prediction F1 score.

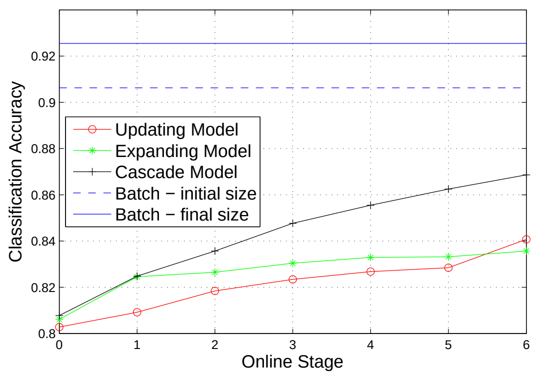


Figure 6: Classification accuracy of online Boosting Tree on defect classification

6.2 Online Boosting Tree for Defect Classification

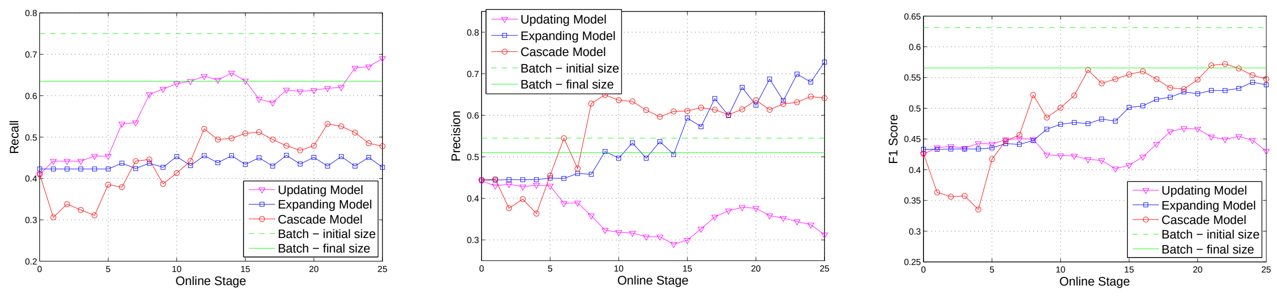
The proposed online supervised and semi-supervised BT model is a general learning technique. Hence, it can be applied to a variety of applications. In this section, we present results of the proposed method being applied to defect cause classification. The goal of defect cause classification is to decide the reason that causes the defective region to happen. Three defect causes are in our consideration: aberrance on TFT, coat and foreign substance. Each defective image should be considered as a sample in this scenario. We collect a group of defect images (130 images in total) whose defect cause has been labeled by operator. These 130 images are then divided into 3 sets: offline training set (15 images), online training set (90 images) which become available at 6 online training stages with 15 samples in each stage, and testing set (25 images).

The classification accuracy is shown in Fig.6, which has been evaluated for “Updating Model” (red), “Expanding Model” (green), “cascade model” (black), “batch model with size of initial model” (dashed blue) and “batch model with size of final model” (solid blue). The accuracy is defined as the percentage of samples of which the classifier provides correct label. The results show that performance keeps increasing as more samples become available. Therefore, our proposed online Boosting tree is not restricted to a specific application; it works well in other tasks.

6.3 Effectiveness of Compact Representation

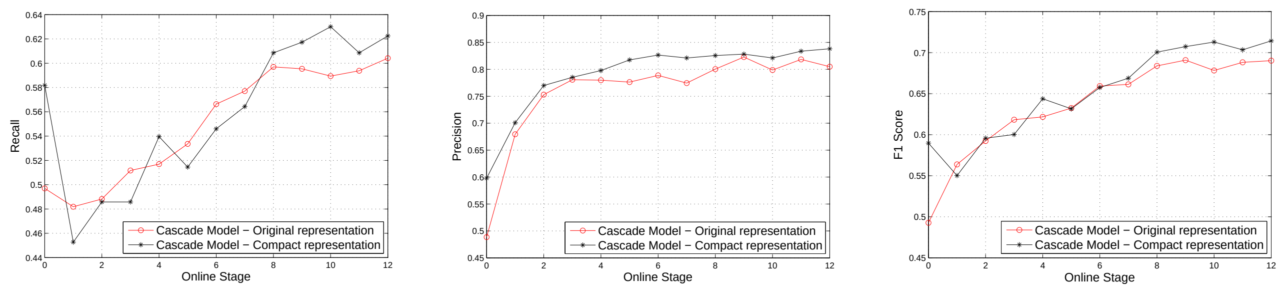
In this section we will show how the compact representation of samples will influence the performance of our proposed algorithm.

The task here is defect detection. A smaller data set was used this time, in which the initial training only used pixels from two images, and 12 images were used for online training with one image added in each online stage. There are another six images used for testing. All these 20 images were randomly selected, and the experiments were performed for 20 rounds. Fig.7 shows the average performance of recall, precision and F1 score in each online stage evaluated on the total 20 images. The red line corresponds to the Boosting tree method which keeps all original samples in memory and



(a) Recall (b) Precision (c) F1 score

Figure 5: Online Boosting Tree on defect detection



(a) Recall (b) Precision (c) F1 score

Figure 7: Online Boosting Tree with compact representation on defect detection

lets it grow as online stage goes on. The black line corresponds to the same method with compact representation, and in all the time of the online training, the sample stored in memory will be kept as a constant size of 20,000.

From the figure, we can see that, with compact representation of appropriate size, the performance doesn't get hurt. Sometimes, compact representation shows better performance than the original representation, that is possibly because the quantization in compact representation will remove some noise in original samples.

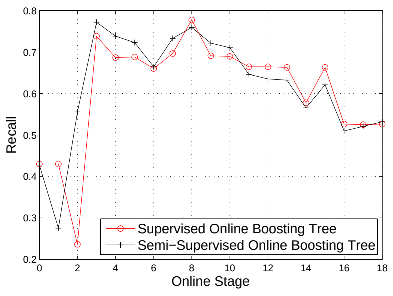
6.4 Semi-Supervised Online Boosting Tree for Defect Detection

In this section, we compare the performance of online supervised BT with online semi-supervised BT. The same training and testing set as Sec.6.3 is used in this analysis. To make a fair comparison between the supervised and semi-supervised algorithm, the images are fed into the algorithm in exactly the same order. However, when an unlabeled image arrives, the supervised algorithm does nothing, and performance is the same as previous stage; the semi-supervised algorithm will perform co-training or self-training. Therefore, there are in total 18 online stages, each stage with input of one image(12 labeled, and 6 are unlabeled).In each co-training step, the first 5,000 samples which satisfy the maximum difference condition are selected as the most confident samples.

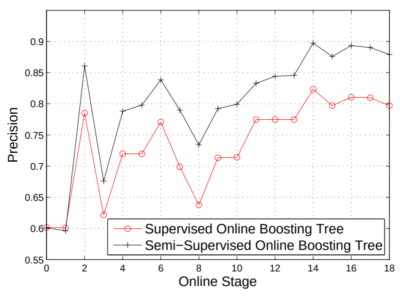
The performance evaluated on the whole data set is shown in Fig.8. The black line corresponds to semi-supervised online Boosting tree model, while the red line corresponds to online supervised model. Semi-supervised learning, in most cases, can help to improve the performance, if the unlabeled samples could reveal the true distribution of the whole data set. Or equivalently, the required number of labeled samples is reduced for achieving comparable performance, which means the label work of human operator will be reduced.

7. CONCLUSION

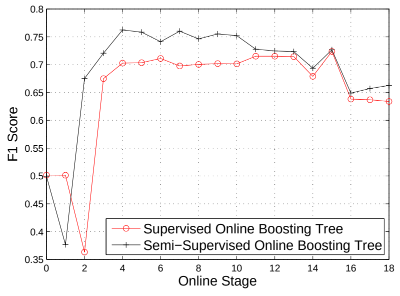
In this paper, we proposed Online Supervised Boosting Tree (OSBT) to adapt and evolve the classifier in an online fashion for industrial machine vision applications. The proposed OSBT prevents drift by introducing a initial prior BT model. During online learning, this initial target model is then updated and expanded to adapt to changing data characteristics. Cascade prediction by combining Updated BT and Expanded BT further increases prediction performance. We also proposed Online Semi-supervised Co-Training (OSSCT) method to enhance performance by utilizing the large amount of unlabeled samples. The Expanded BT and Updated BT co-train each other as two complimentary clarifies which removes the constrains of independent views. To cope with memory growth, we proposed compact representation of the data where the knowledge representation remains compact and free from redundancy to fit into the limited memory, and the algorithm maintains a fixed computational complexity. We demonstrate the efficacy of our algorithms on two industrial inspection tasks: defect detection & defect classification for LCD TV panels. Results on hundreds



(a) Recall



(b) Precision



(c) F1 score

Figure 8: Semi-supervised online Boosting Tree on defect detection

of LCD TV images show that the cascade classifier achieves good balance between precision and recall, leading to highest F1 score, and our semi-supervised co-training scheme increases prediction performance over the online supervised BT by approximately 4%.

8. REFERENCES

[1] S. Avidan. Ensemble tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 494–501, 2005.

[2] B. Babenko, M.-H. Yang, and S. Belongie. Visual Tracking with Online Mulitple Instance Learning. In *Proc. CVPR*, 2009.

[3] A. Blum and T. Mitchell. Combining Labeled and Unlabeled Data with Co-training. In *Proc. of the 11th Annual Conf. on Computational Learning Theory (COLT)*, pages 92–100, 1998.

[4] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] O. Camoglu, T. Yu, L. Bertelli, D. Vu, V. Muralidharan, and S. Gokturk. An Efficient Fashion-driven Learning Approach to Model User Preferences in On-line Shopping Scenarios. In *Proc. Online Learning for Computer Vision Workshop, in conjunction with CVPR 2010*, pages 28–34, 2010.

[6] G. Cauwenberghs and T. Poggio. Incremental and Decremental Support Vector Machine Learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[7] Y. Freund and R. Schapire. A Decision Theoretic Generalization of On-line Learning and an Application to Boosting. *J. of Computer and System Sciences*, 55(1):119–139, 1997.

[8] J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: A Statistical View of Boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[9] F. Gayubo, J. Gonzalez, E. Fuente, F. Miguel, and J. Peran. On-line Machine Vision System for Detect Split Defects in Sheet-metal Forming Processes. In *Proceedings of the 18th International Conference on Pattern Recognition*, 2006.

[10] A. Goldberg, M. Li, and X. Zhu. Online Manifold Regularization: A New Learning Setting and Empirical Study. In *Proceeding of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.

[11] H. Grabner and H. Bischof. On-line Boosting and Vision. In *Proc. of IEEE Computer Vision and Pattern Recognition*, 2006.

[12] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised On-line Boosting for Robust Tracking. In *Proc. ECCV*, 2008.

[13] L.-J. Li, G.Wang, and L. Fei-Fei. Optimol: automatic online picture collection via incremental model learning. In *Proc. CVPR*, 2007.

[14] X. Liu and T. Yu. Gradient Feature Selection for Online Boosting. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[15] S. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[16] E. Lughofer. On-line Evolving Image Classifiers and Their Application to Surface Inspection. *Image and Vision Computing*, 28(7):1065–1079, 2010.

[17] V. Nair and J. J. Clark. An Unsupervised, Online Learning Framework for Moving Object Detection. In *Proc. CVPR*, 2004.

[18] F. Orabona, C. Castellini, B. Caputo, J. Luo, and G. Sandini. Indoor Place Recognition Using Online Independent Support Vector Machines. In *18th British Machine Vision Conference (BMVC)*, 2007.

[19] N. Oza and S. Russell. Online Bagging And Boosting. In *Proc. Artificial Intelligence and Statistics*, pages 105–112, 2001.

[20] A. Pronobis, L. Jie, and B. Caputo. The more you learn, the less you store: Memory-controlled incremental SVM for visual place recognition. *Image and Vision Computing*, 28(7):1080–1097, 2010.

[21] J. R. Quinlan. Introduction to Decision Trees. *Machine Learning*, 1(1):81–106, 1986.

[22] F. Tajeripour, E. Kabir, and A. Sheikhi. Fabric Defect Detection Using Modified Local Binary Patterns. *EURASIP Journal on Advances in Signal Processing*, 2008.

[23] P. Utgoff, N. Berkman, and J. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1):5–44, 1997.