Human–Machine Interaction Issues in Quality Control Based on Online Image Classification

Edwin Lughofer, James E. Smith, Muhammad Atif Tahir, Praminda Caleb-Solly, Christian Eitzinger, Davy Sannen, and Marnix Nuttin

***Abstract*—This paper considers on a number of issues that arise when a trainable machine vision system learns directly from humans. We contrast this to the "normal" situation where machine learning (ML) techniques are applied to a "cleaned" data set which is considered to be perfectly labeled with complete accuracy. This paper is done within the context of a generic system for the visual surface inspection of manufactured parts; however, the issues treated are relevant not only to wider computer vision applications such as medical image screening but also to classification more generally. Many of the issues we consider arise from the nature of humans themselves: They will be not only internally inconsistent but also will often not be completely confident about their decisions, particularly if they are making decisions rapidly. People will also often differ systematically from each other in the decisions they make. Other issues may arise from the nature of the process, which may require the ML to have the capacity for real-time online adaptation in response to users' input. Because of this, it may be that the users cannot always provide input to a consistent level of detail. We describe how all of these issues may be tackled within a coherent methodology. By using a range of classifiers trained on data sets from a compact disc imprint production process, we present results which demonstrate that training methods designed to take proper consideration of these issues may actually lead to improved performance.**

***Index Terms*—Human–machine interaction (HMI), image classification, insight into classifier structures, online adaptation, partial confidence, resolving contradictory inputs, variable input levels.**

I. Introduction

In MANY machine vision applications, such as inspection tasks for quality control, an automatic system tries to reproduce human cognitive abilities. Even if a human expert is told to apply a set of well-defined rules, a lot of subjective past experiences will be incorporated in the decision. Any machine vision system that does not consider such an experience will fail to achieve a high classification accuracy. The most efficient and flexible way to transfer this experience into the software of a machine vision system is to learn the task from a human expert [9]. Traditionally, this is done either by an expert providing preclassified images for supervised learning or by knowledge acquisition from human operators in the form of rule bases. Typically, machine learning (ML) systems are trained in a supervised batch mode from a set of example data items, each of which has a unique label. Although there may be inconsistencies or noise within the data, these are generally considered to be random in nature, and each point is considered to be labeled with complete accuracy.

However, as ML technology moves from research laboratories to practical applications such as machine vision, a range of issues arise concerning how humans relate to and interact with such systems [11], [21]. Not only does this question the feasibility or even relevance of considering "cleaned" data sets, there is an increasing demand for systems to operate in situations where offline batch-mode processing is not appropriate [18]. This can occur if data are hard, time consuming, or costly to obtain, or if the underlying processes change fairly rapidly, requiring reconfiguration. Both of these cases lead to the need for an element of incremental online training [23], [44]. Nevertheless, this prompts a renewed interest in the *nature* of the human interaction with adaptive ML systems [2], [6].

In this paper, we focus on a number of issues relating to human–machine interaction (HMI) in the context of a generic system for the visual surface inspection of manufactured parts. A big challenge in the design of HMI scenarios is that they can be handled by the expert in the system domain, without the necessity of being experts in computer science [3], [12]. This property makes them applicable to (a wider range of) end users. The issues proposed in this paper fulfill this strong property as embedded in a quality control system the system experts and end users work with regularly. Section II describes the basic architecture of the proposed system, and Fig. 1 shows the impact of the issues therein. Following a description of the data sets (Section III), the HMI issues are considered as follows.

1) Section IV deals with the issues arising when the nature of the application demands real-time online learning after an initial batch-mode phase (HMI 1). We show how classifiers which can be trained incrementally can outperform static ones, which cannot be further refined in response to incoming data.

Manuscript received March 14, 2008; revised October 18, 2008. First published August 7, 2009; current version published August 21, 2009. This work was supported in part by the European Commission Project (Contract STRP016429, acronym DynaVis) and in part by the Upper Austrian Research Promotion. This paper was recommended by Associate Editor L. Rothrock.

E. Lughofer is with the Department of Knowledge-based Mathematical Systems, Johannes Kepler University of Linz, 4040 Linz, Austria (e-mail: edwin.lughofer@jku.at).

J. E. Smith and P. Caleb-Solly are with the Bristol Institute of Technology, University of the West of England, BS16 1QY Bristol, U.K. (e-mail: james.smith@uwe.ac.uk; praminda@uwe.ac.uk).

M. A. Tahir is with the Centre for Vision, Speech and Signal Processing, University of Surrey, GU2 7XH Surrey, U.K. (e-mail: muhammad.tahir@uwe.ac.uk).

C. Eitzinger is with the Profactor GmbH, 4407 Steyr, Austria (e-mail: christian.eitzinger@profactor.at).

D. Sannen and M. Nuttin are with the Department of Mechanical Engineering, Division PMA (Production engineering, Machine design and Automation), Katholieke Universiteit Leuven, 3001 Leuven, Belgium (e-mail: davy.sannen@mech.kuleuven.be; marnix.nuttin@mech.kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCA.2009.2025025

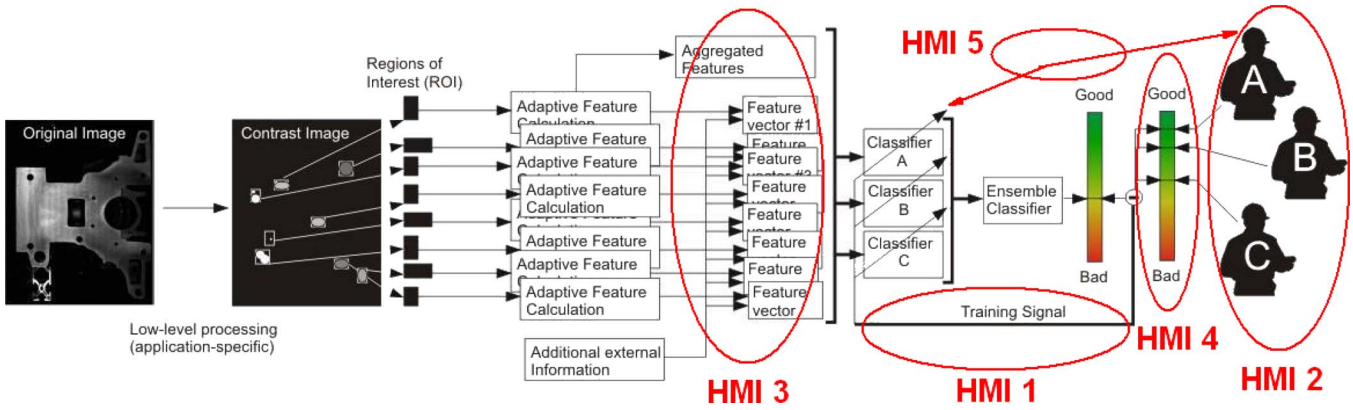

Fig. 1. Classification framework for classifying images into good and bad, with the five major HMI issues marked with red ellipsoids.

2) Section V considers the fact that different users will often differ systematically from each other (HMI 2) and how best to incorporate this diversity of information. The differences are influenced by the reliability of the users, which can be caused by skill, mood, weather conditions, time pressure, etc. [47].
3) Section VI considers how the demand for rapid responses may reduce the level of detail in the feedback that users can produce (HMI 3), and suggests ways for dealing with this.
4) Section VII considers the fact that, for a number of reasons, the operator(s) may not be completely confident in their decisions (HMI 4), and shows how an enhanced HMI interface for online labeling can be exploited to capture this information and lead to performance improvements.
5) Section VIII issues related to the interpretability of the classifiers are considered (HMI 5). This may motivate the operators to interact with the system on a system model level.

All these scenarios support the interaction with the operators in a direct way rather than performing a complex information analysis as, for example, done in [1]. In fact, operators have the option to provide direct and more detailed information than just a simple go/no-go sign. Any further information from the operators will guide the whole quality control system to a higher accuracy (as empirically shown in the subsequent sections).

II. Generic Architecture for Image Classification

The whole framework is shown in Fig. 1. Starting from the original image, a "contrast image" is calculated. A predefined master image is used as an ideal and fault-free reference situation. Each red/green/blue pixel value of a newly recorded image is compared with the corresponding (same $x-y$ coordinate position) red/green/blue pixel value of the master image plus–minus a threshold, which may vary for different regions in the image (e.g., in homogeneous regions typically lower than in edgy regions). These thresholds are preestimated based on historic data and long-term experience. The absolute values of deviation in red, green, and blue are averaged to an overall deviation in gray value (ranging from 0 to 255). In this sense, the gray value of each pixel represents the degree of deviation from the normal appearance of the surface. For all further discussions, we disregard these low-level processing steps and assume that an appropriate and stable system of image acquisition, preprocessing, and image segmentation is in place. In particular, we assume that enough relevant information is captured to permit correct decisions to be made purely based on each image.

From the contrast images, regions of interest (ROIs) are extracted, each of which contains a single object which may or may not be a fault. Various ROI extraction methods for grouping local and similar data clouds were applied, and their accuracy was tested, ranging from connected components, morphological approaches to clustering approaches for grouping such as iterative k-means, hierarchical clustering [22], reduced Delaunay graph [35], DBSCAN algorithm [15] etc. For each ROI, a total of 57 object features are calculated, such as area, brightness, homogeneity, or roundness of objects, characterizing their shape, size, etc. These are complemented by aggregate features which characterize images as a whole, such as the number of objects, the maximal density of objects, or the average brightness in an image. The full list of aggregated features is shown in Table I. This list was derived on the basis of discussions with experts on surface inspection and wider machine vision applications. Some features appear as quite intuitive to have a high discriminative power, particularly the number and total area of all ROIs (typically, the more deviating pixels the contrast image show, the higher the likelihood that the corresponding production item is bad) or the maximal/average local density of ROIs (dense regions are more likely to belong to bad parts). Depending on the nature of the operator's feedback, the data describing individual ROI may be added to the aggregate image data in various ways.

The feature vectors are then processed by a trained classifier system that generates a final good/bad decision for the whole image. We have implemented and evaluated many different classifiers within this framework, and the "best" classifier is, of course, problem dependent [53]. Here, we focus on reporting the results of three classifiers which performed well on these data sets. For offline training, these were two decision tree-based classifiers *CART* [5] and *C4.5* [37], along with k-Nearest Neighbors (*kNN*) [20]. These methods are widely known and used and were recently named among the top 10 data mining algorithms [55]. We also applied two incremental learning algorithms, i.e., *eVQ-Class* [29] and *FLEXFIS-Class* [31]. Because the framework is generic, it produces many features

TABLE I
LIST OF "AGGREGATED" IMAGE-LEVEL FEATURES USED WITHIN GENERIC FRAMEWORK

No.	Description	No.	Description
1	Number of ROIs	10	Max. grey value in the image
2	Average minimal distance between two ROIs	11	Average grey value of the image
3	Minimal distance between any two ROIs	12	Total area of all ROIs
4	Size of largest ROI	13	Sum of grey values of all faults
5	Center Position of largest ROI, x-coord	14	Maximal local density of the ROIs
6	Center Position of largest ROI, y-coord	15	Average local density of the ROIs
7	Max. intensity of all ROIs	16	Average area of the ROIs
8	Center Position of ROI with max. intensity, x-coord	17	Variance of the area of ROIs
9	Center Position of ROI with max. intensity, y-coord		

describing each ROI, not all of which may be relevant to any given task. To reduce the "curse of dimensionality" effect [20], we therefore applied Tabu Search to perform feature selection as described in, e.g., [4] and [48].

The classifiers are trained using operator input. For a limited period of time, the operators judge the parts in parallel to the machine vision system. They usually inspect the real part (not the image) and make a decision, which is then fed back into the classifier. In the simplest case, the operator just pushes a green ("*good*") or red ("*bad*") button. If the speed of the production process allows, they may also provide more detailed training input (this will be treated in HMI issues #3 and #4, i.e., in Sections VI and VII). For many reasons, it may be desirable to aggregate input from different operators who are separated in time (shift patterns. . .) or space (multiple lines or sites). To cope with this, reflect different opinions, and maintain user engagement, a separate classifier is trained using the input from each operator. When these trained classifiers are applied to new images, any contradicting decisions are resolved using an ensemble (more precisely, classifier fusion) method [25] to provide the final decision of the system (see Section V, HMI issue #2). In addition, there might be a quality control supervisor, which does not do the daily quality inspection him-/herself, but who wants to be in control as much as possible. If the supervisor also labels (part of) the data, the system will take into account this information, as well, by using it for training the ensemble method in such a way that its output models the supervisor's decisions.

In general, classification performance will be measured in terms of tenfold cross-validation (CV) error, except for the incremental online training issue, where the accuracy on a separate test set is calculated (as CV is an offline procedure). Sometimes, this whole training process may fail to produce satisfactory results, most often because there is important information missing in the image. This may be due to inadequate lightning or failure of the image segmentation. To detect such situations, we have developed an early warning system that predicts the success or failure of the training process [45].

III. CHARACTERISTICS OF THE CD IMPRINT DATA SET

The whole recorded data set contains 1687 images, from which 153 contrast images are fully black, hence containing no potential fault candidates on it and can be classified directly as good. The images were labeled by four different operators, which assigned the labels not only to the whole images but also to each single ROI. A graphical user interface (GUI) was designed (see Fig. 2) in consultation with the users to allow them to rapidly assign both a class (0–12) (6 belonging to

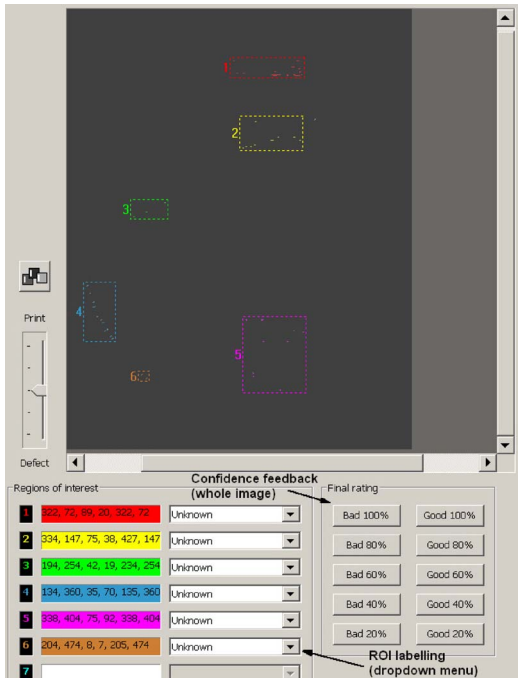

Fig. 2. GUI of labeling tool. Here, six ROIs are found which can be all labeled by a drop-down control element (lower left), and image labels can be assigned together with a confidence (lower right).

pseudoerror = no real errors and 6 belonging to real errors) and a severity to each object segmented in the image. In these images, the class distributions were not well balanced; in this sense, it was not possible to train a high-performance stand-alone object classifier. However, when using their output information as additional features (under the scope of HMI 3, according to Fig. 1), we could improve the classification as will be demonstrated in Section VI.

Examples of faults on compact discs (CDs) caused by the imprint system are a color drift during offset print, a pinhole caused by a dirty sieve (color cannot go through), occurrence of colors on dirt, and palettes running out of ink. These have to be distinguished between so-called pseudofaults, for instance, shifts of CDs in the tray (disc not centered correctly), causing longer arc-type objects (e.g., see the upper left ROI in Fig. 3) or masking problems at the edges of the image or illumination problems (causing reflections). This leads to quite complex structures in the deviation images, ranging from blob-type pixel clouds over arc-type objects (usually pseudoerrors) to large widespread pixel areas (usually stemming from dirt and palettes running out of ink). For an image example, see Fig. 3: Different types of script nameplates (at different positions) form separate ROIs. In addition, the different arc-type objects (two in the

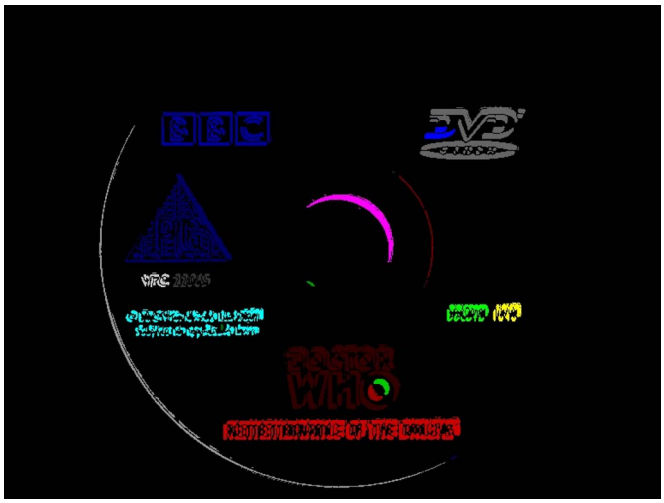

Fig. 3. Typical example of a contrast image from the CD imprint production process. Different ROIs found by DBSCAN are marked with different gray levels/colors.

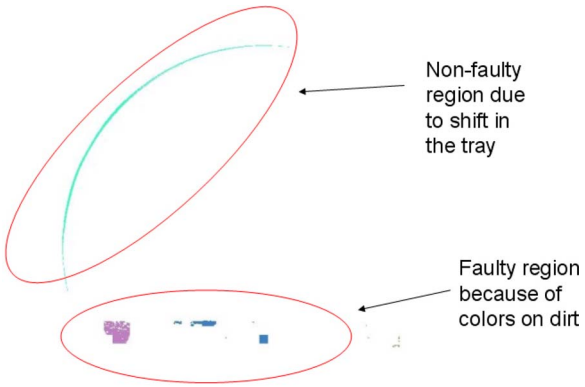

Fig. 4. Deviation image from the CD imprint process. Different colors/gray levels represent different ROIs. Note that most of the image is white, which means that the deviation to the fault-free master is concentrated in small regions; the faulty and nonfaulty parts are exclusively marked as such.

middle and one long one outside partially circumventing all other objects) (both marking pseudoerrors as due to shift of the CD in the tray) and the small but clearly visible blob in the middle of the image (a real error) all represent different ROIs; the different gray levels (respectively, colors) represent a grouping achieved by the DBSCAN algorithm [15]. In Fig. 4, another example of a deviation image is presented from the CD imprint production process: Different ROIs found by the ROI recognition method are highlighted in different colors (respectively, gray levels). In fact, the longer arc-type object (in light gray) denotes a completely other type of error than the two rectangle and compact type of regions (in darker gray levels). An interesting point here is that on the right most part of the faulty region (marked with blue pixels) would be not enough to classify the whole image as "bad," whereas the whole region is significant enough to classify the image as "bad." This means that aggregated features explaining the intensity and area of the pixel clouds may be necessary for discriminating between "good" and "bad" images. However, this is not sufficient, as it also depends on the shape characteristics of the ROIs: Imagine two arc-type objects with the same intensity and area as the faulty region shown in Fig. 4. This image would be classified as "bad," although containing two pseudoerrors. In this sense, it is important to add object features characterizing shape, density, etc., of single objects either supervised or unsupervised to the aggregated features. The class distributions of the labeled ROI were not well balanced, which meant that it was not possible to train a high-performance stand-alone object classifier; however, when using their output information as additional features (under the scope of HMI 3, according to Fig. 1), we could improve the classification as will be demonstrated in Section VI.

Applying these classification algorithms to the standard aggregated feature sets (containing 17 predefined features) of real-world data from an online CD imprint production process produced accuracies between 90% and 94% [42] within a tenfold CV procedure [46]. Even though the accuracies lie in a reasonable range, they fall far below the standards needed for automated industrial quality control or desirable for medical diagnostic support (often, $> 96\%$ is required). However, as the following sections detail, it is possible to gain significant improvements by considering the human aspects of the sources of data from which the ML system learns. Another benefit is that the resulting systems are also more transparent, user friendly, and generally applicable.

IV. Incremental Classifiers Based on Operators' Feedback

Incremental training involves the adaptation of parameters and evolving structures (e.g., evolving neurons, rules, etc.) and is required whenever the operator gives a feedback upon the classifier(s) decisions during online production mode. This is because a periodic rebuilding of the classifier using all the samples seen so far is impractical as it slows down the training process too much. On the other hand, if the classifier(s) are not updated at all (i.e., they are kept fixed after the initial offline batch training phase), they cannot refine their parameters, react to changing operating conditions or system behaviors, and hence may result in unsatisfactory performance. For dealing with the online learning problem, we exploited an evolving clustering-based classifier (*eVQ-Class* [29]) and an evolving fuzzy rule-based classifier (*FLEXFIS-Class* [31]). The first one is a clustering-based classifier based on vector quantization [19] and exploits the idea of a vigilance parameter motivated in [8] for forming an incremental and evolving variant. It is incremental in the sense that the cluster centers and surfaces are updated samplewise based on new incoming samples during the online process. It is evolving in the sense that new clusters are evolved based on Mahalanobis distances from a newly loaded sample (feature vector extracted from a newly recorded image) to all the clusters: The minimal value of these Mahalanobis distances is compared with a threshold, also called vigilance parameter, and if greater than this, a new cluster is born, otherwise the nearest cluster is updated. In this sense, the latter controls the tradeoff between stability (i.e., the generation of new clusters) and plasticity (i.e., the updating of existing clusters). The cluster prototypes (centers) are updated by shifting them toward new incoming samples, using a learning rate which starts with value 0.5 and decreases with the size of the clusters, i.e., by $0.5/k_i$, with k_i being the number of samples forming the cluster i, i.e., those samples for which this cluster was the nearest one. The surfaces of the ellipsoidal clusters are estimated by the

width of the cluster in each dimension and synchronously updated by exploiting a recursive variance formula [36]. Another modification of the conventional vector quantization approach is that to find the winning cluster: It does not calculate the distance of a new incoming sample to all the cluster centers but to the ellipsoidal surfaces of the clusters. If a new sample falls inside some ellipsoids, the distances to the center of these clusters are taken and the winning cluster is elicited. This cluster is then updated, and in this case, a new cluster is never born. Each cluster is labeled with the class most frequently represented among the samples it contains (for further details, see [29]).

The second incremental classifier evolves multiple Takagi–Sugeno fuzzy (regression) models [50] and a one-versus-rest classification scheme [13] in case of $K > 2$ classes. For each class, a separate Takagi–Sugeno fuzzy model is trained based on an indicator matrix, obtained by setting the label entries for the current class to one and for all others to zero. The final multimodel fuzzy classifier is extremely flexible for integrating new classes on demand as they arise, user defined, during the online production and learning process. As fuzzy models are highly nonlinear and hence able to force the regression surface of a model to go toward zero more rapidly, as compared to inflexible linear hyperplanes, the masking effect (which is a severe problem in the case of linear regression by indicator matrix [20]) is much weaker than in the pure linear case. The single Takagi–Sugeno fuzzy models are evolved by the FLEXFIS (FLEXible Fuzzy Inference Systems) approach [30], which connects the eVQ [29] for evolving the fuzzy rules and for incremental training of the antecedent parts with the recursive least squares approach [28] for consequent adaptation in a way, such that near optimality in the least squares sense is achieved. In these terms, we speak about a "robust" incremental learning approach coming close to the hypothetical solution if all online samples would have been sent into a batch learning process.

The methods are used to generate initial offline batch trained classifiers, which are further updated and evolved as new samples arrive. However, it is necessary to alleviate the burden of input on the user, by reducing the number of responses they are required to give. Most so-called active learning approaches (e.g., see [51]) accomplish this by only querying the user if the system's confidence in its own decision is below some threshold. However, this can be a risky strategy in situations where examples of previously unseen classes may arise or if the user wishes to override previous decisions, which could happen for a range of human- or production-related reasons. The approach that we have taken is as follows.

1) The system predicts the class of each new sample.
2) This is shown visually to the user.
3) The default position is to assume the user's tacit agreement.
4) In this case, the system is updated using its own decision output and taking into account the effect on class imbalance—for example, a maximal imbalance of one-third and two-third is allowed in 0/1 case.
5) If the user disagrees with the prediction, they can provide contradictory input. In this case, the system is updated with this input as a training signal.

TABLE II
PERFORMANCE OF INCREMENTAL ONLINE VERSUS STATIC (AND RETRAINED) BATCH CLASSIFIERS

Dataset	Op01	Op02	Op03	Op04	Op05
CART static	70.94	78.82	80.00	76.08	68.2
CART retrained	82.74	90.59	90.39	88.24	81.7
eVQ-Class static	68.82	76.67	76.27	74.71	70.0
eVQ-Class inc.	83.33	88.82	88.43	88.43	82.0
FLEXFIS-Class static	68.63	85.29	85.68	78.24	74.6
FLEXFIS-Class inc.	84.12	88.06	89.92	84.90	84.2

From an HMI perspective, this changes the way that the question is posed to the user: The system is effectively asking the user "*Am I right that this a type X?*" as opposed to the typical "*what type is this?*."

N-fold CV assumes a fixed data set and so is not an appropriate measure here. Instead, the CD imprint data set was split into three. The first third of the images is used for initial offline training. The middle third is used to simulate incremental online training of the classifier and is sent sample per sample into *eVQ-Class* and *FLEXFIS-Class*. The final third is used as a test set for evaluating the trained classifiers. This is an appropriate way of estimating the true online accuracy as the whole CD imprint data were recorded and stored online, so the test samples do represent the most recent images. Table II shows the performance of the incremental classifiers versus their corresponding batch versions, i.e., trained in initial offline mode with the first batch of data and not further updated (kept static): It can be seen that by doing an adaptation during online mode, the performance on new unseen samples significantly increases by 10%–15% for all operators. Furthermore, Table II also shows us that, when retraining CART in batch mode on all training samples, the accuracy on the new unseen samples is only marginally higher than for the incrementally trained approaches; in this sense, the computationally intensive retraining does not really pay off. Because we have used a different method for estimating the classifier accuracy, it is not possible to directly compare the exact values, but it is interesting to consider the effect of switching from shuffled data sets (used in N-fold CV), where all classes may be represented, to time-ordered data, where some operators perceive that certain classes of defects only occur after a certain time. Closer inspection shows that, for those operators such as Operator 1 where there is a time imbalance in the class distribution, the gap between the "shuffled" results and the "time-ordered" ones is less for the incrementally adapting *eVQ-Class* than it is for even the retrained CART. From an HMI perspective, online learning highlights some interesting points. In a batch-mode model, the user is asked to perform repeated interactions for image labeling without feedback or reward. This can make the process seem time consuming and possibly pointless. In contrast to this, online training means that the user can "see" the system learning from their input, building a progressively more accurate model, which can help to motivate them and increase their focus and attention. Similar findings have been found in a range of other HMI studies (e.g., see [39]).

V. HANDLING INPUT FROM MULTIPLE USERS

When learning systems are applied to industrial applications and the operators are actively involved in the training

process, the fact that usually multiple operators will be working on the system needs to be taken into account. Similarly, in the fields of medicine and science, it may be desirable to incorporate inputs from a wide range of people in order to broaden the range of data examples and human perspectives available. Inevitably, every time a person makes a decision, they implicitly consider and weight multiple competing criteria such as the desire to reduce the risk by wrongly classifying "defective" samples as "OK," or vice versa. One approach to this problem is to combine the inputs and train multiple versions of classifiers with different parameter and then select a global "winner" (e.g., by inspecting the receiver operating characteristic curves or by a systematic fuzzy decision-making process [43]). However, this still assumes a consistent behavior between and within user's inputs. It also assumes a fixed weighting of each user's inputs. In the proposed framework, we take a different approach based on the assumption that some stage decisions will be taken to define, e.g., company policy or diagnostic standards and that these can be captured as a set of examples labeled by some "expert." In the first stage of our system, each operator trains his own personal classifier in the way he thinks is the best. In the second stage, the outputs of these individual classifiers are then combined into the final decision. Note that this approach could, in principle, work equally well whether the individual classifiers are trained by the users' labeling the same or disjoint sets of examples.

The idea of *classifier ensembles* is to train not one but a set (ensemble) of classifiers. Most research has considered the case where there is only a single data set for which a diverse set of classifiers is trained. Here, these ensemble methods are used in a different context: They are used to combine the outputs of classifiers which are trained by different operators (i.e., using different training sets). This means that the ensembles are used to resolve the contradictions between the operators. Two levels of contradiction among the operators can be distinguished.

1) *Interoperator contradictions*: systematic contradictions between the decisions of different operators. They can be caused, e.g., by different levels of experience, training, skill, etc.
2) *Intraoperator contradictions*: contradictions (often more random) between the decisions of a single operator. They can be caused by "personal" factors, such as the level of fatigue, attention, stress, boredom, etc., or "environmental" factors, such as a changed company policy concerning the quality control, recent customer complaints, etc.

The intraoperator contradictions, which are mostly random, are dealt with by the operators' own classifiers themselves. Several learning techniques can naturally handle noisy data (e.g., see [13]). To handle the systematic interoperator contradictions, the operators train their own classifiers, the outputs of which are combined by an ensemble method. An advantage of having each operator train his "own" classifier is that these classifiers will be easier to train (provided that sufficient data are available), since only the intraoperator contradictions need to be handled by these classifiers. Furthermore, it is clear what has been taught to the system by which operator, making it possible to provide operator-specific feedback.

Ensemble methods can be divided into two classes: generative and nongenerative [52]. *Generative* ensemble methods generate sets of classifiers by acting on the learning algorithm or on the structure of the data set, trying to actively improve the diversity and accuracy of the classifiers. These include well-known methods such as "Boosting" [16]. In general, these algorithms build different classifiers iteratively, with each iteration focusing on the data points that were hard to classify by the classifier in the previous iteration by weighing them appropriately. When a new data item is to be classified, the decisions of the classifiers from the different iterations are combined using a weighted majority vote. This kind of procedure is not applicable here since there is a fixed set of classifiers (exactly one per operator). These classifiers cannot be modified by the ensemble algorithm (as they represent the decisions of an operator), and the ensemble method cannot create additional classifiers.

In contrast, *nongenerative* ensemble methods do not actively generate new classifiers but try to combine a set of different existing classifiers in a suitable way. Clearly, the nongenerative approach is necessary for the application in this paper: The operators train their own classifiers in the way they think is the best, and there is no way for the system to intervene in this process. Nongenerative ensembles can be generally divided into two groups: classifier selection and classifier fusion [54]. The assumption of the former is that each classifier is "an expert" in some local area of the feature space. The latter assumes that all classifiers are trained over the whole feature space. For the application in this paper, classifier fusion is more appropriate, since the operators train the system with the data which are provided by the vision system, which could be spread over the entire data space. The fusion of the outputs of the different classifiers (trained by the different operators) can be done using fixed or, if a "supervisor" has labeled the data, trainable classifier fusion methods. In the latter case, the ensembles are optimized to best represent the decisions of the supervisor (i.e., these data are considered to be the ground truth), taking into account only the decisions of the operators. The fixed classifier fusion methods include *Voting* methods [27] and algebraic connectives such as *maximum*, *minimum*, *product*, *mean*, and *median* [24]. Trainable classifier fusion methods include the *Fuzzy Integral* [10], *Decision Templates* [26], *Dempster-Shafer combination* [40], and *Discounted Dempster-Shafer combination* [41]. In addition, the *Oracle*, a *hypothetical* ensemble scheme that provides the correct result if at least one of the classifiers in the ensemble outputs the correct classification, was considered. The accuracy of the Oracle can be seen as a "soft bound" on the accuracy, which can be achieved by the classifiers and classifier fusion methods (for a detailed survey of these methods, see for example, [25]). We also considered *Grading* [17], a different approach to classifier fusion. Here, a so-called "Grading" classifier is trained for each base-level classifier to predict whether the classifier will be correct for a given data sample. The classifiers, which are estimated to be correct, are considered in a majority voting scheme to provide the final output. Thus, Grading forms a nonlinear weighting over the feature space for each classifier. In this sense, it could be considered akin to the selection methods for classifier fusion. The difference is that, in this case, the weighting implicitly captures the preferences and opinions of the superoperator in terms of conflicting risks, etc. Thus, in different regions of the feature space, it will adapt to give

TABLE III
CLASSIFICATION ACCURACY (IN PERCENTAGE) OF CART CLASSIFIERS FOR THE CD DATA SETS. CLASSIFIERS ARE TRAINED USING DATA FROM ONE OPERATOR (ROW) AND EVALUATED FOR THE ABILITY TO PREDICT LABELS PROVIDED BY ANOTHER (COLUMN). BOLD TYPE INDICATES CLASSIFIERS TESTED AND TRAINED ON SAME DATA SET

Classifier trained on data from	Evaluation on data from (supervisor)				
	Op01	Op02	Op03	Op04	Op05
Op01	**90.57**	89.27	85.33	88.94	71.42
Op02	88.83	**95.38**	91.88	93.10	69.42
Op03	86.50	93.42	**93.90**	92.68	70.59
Op04	88.82	93.67	92.08	**94.38**	71.91
Op05	72.42	71.64	71.73	73.58	**91.25**

more weight to those classifiers representing operators whose decisions and weights agree with the superoperator.

The CD data with 17 aggregate features described in Section II were labeled by four different operators. To investigate the HMI issues arising from the interoperator contradictions, a fifth operator (Operator05) was included in the tests, who has a different position in the company than the other operators and thus has a different subjective view on the quality criteria, as will be demonstrated next. As Operator05 is not involved in the day-to-day quality control, this operator is not considered in the other sections of this paper (describing the other HMI issues—not considering the effect of training the system using the inputs of multiple operators). To investigate the effect of the interoperator contradictions, each of these five operators is considered as the "supervisor" in turn. Classifiers are trained for the other operators, and their outputs are then combined by the ensembles in order to better model the decisions of the supervisor. Note that this implies evaluating the classifiers on data which they were not trained for.

In Table III, the effect of training a classifier with the input from one operator (row) and then evaluating this classifier using the input from another operator (column) is shown. It can be seen that three operators make very similar decisions (Operators 02, 03, and 04), one operator differs slightly from these three operators (Operator01), and one operator makes decisions which are very different from all other operators (Operator05). Note that the results of about 90%–95% on the diagonal of this table (shown in bold) denote the evaluation of the classifiers on the same data they were trained on (i.e., a normal classification task) as shown in Section III.

In Table IV, the effect of training a classifier for four operators and then combining them using different ensemble methods to predict the labels provided by the fifth operator (considered as the supervisor) is shown. From these results, we can see that the ensembles are able to represent the supervisor better than the classifiers trained by the different operators themselves. The improvements range up to approximately 3%. In these experiments, the Grading ensemble method performed the best, outperforming the other classifier fusion methods (both fixed and trainable) in all experiments, except when Operator05 is considered as the supervisor (in this case, there is a marginal difference with the Decision Templates method of 0.06%. Note that, if all operators do not agree well with the supervisor, a drop in the accuracy is recorded—e.g., approximately 20% of the decisions of Operator05 (the operator who has a different position in the company than the other operators) do not agree with *any* of the other operators. In this case, even the hypothetical Oracle does not perform well, bounding the achievable accuracy below 80%. From these results, we can conclude that the ensemble methods can effectively be used to combine the decisions of different operators, in order to model the decisions of a supervisor better, with improvements of up to 3%.

TABLE IV
CLASSIFICATION ACCURACY (IN PERCENTAGE) OF THE DIFFERENT ENSEMBLE METHODS (ROWS) FOR THE CD DATA SETS. DIFFERENT OPERATORS ARE CONSIDERED TO BE THE SUPERVISOR (COLUMNS); TRAINING INPUT FOR THE ENSEMBLES IS PROVIDED BY THE OTHER OPERATORS

Ensembles to combine the operator classifiers	Evaluation on data from (supervisor)				
	Op01	Op02	Op03	Op04	Op05
Voting	88.35	92.31	90.22	92.72	70.33
Algebraic Mean	88.19	89.86	88.28	91.56	71.74
Fuzzy Integral	88.50	94.42	91.61	93.69	71.25
Decision Templates	88.19	94.42	89.16	91.59	74.75
D-S Combination	88.19	94.42	91.82	92.07	71.75
Disc. D-S Combination	88.31	94.42	92.24	93.54	71.75
Grading (CART)	91.19	96.05	93.57	94.61	74.69
Oracle	95.83	98.89	97.07	98.60	78.90

Another interesting result is that by combining the outputs of the classifiers trained for the different operators and combining them by an appropriate ensemble method (in this case, the Grading ensemble); in several cases, the performance is even higher than the performance of a classifier specifically trained on the labels provided by the supervisor (the results on the diagonal of Table III). This is caused by the interoperator contradictions among the operators, creating a diverse ensemble of classifiers which is able to perform well, as shown in these results. The only experiment in which this was not the case is when Operator05 is considered to be the supervisor (again due to the large deviations with *all* of the other operators).

VI. HANDLING VARIABLE LEVELS OF DETAIL IN USER INPUTS

A major problem of image classification is the fact that it is not known in advance how many ROIs may be segmented from images occurring in the future, and yet, most classification algorithms assume a fixed-size input data space.

As noted earlier, for each segmented ROI, we calculate a fixed number of features (57). The most straightforward way to tackle this issue is to preprocess the ROI and characterize its distribution within the object feature space by a fixed number of descriptors. In practice, this requires first a preprocessing stage prior to training the image-level classifier and then also during testing/validation, an additional step to process the information about the ROI within each image. When operator-assigned labels are present, supervised object-level classifiers can be constructed [7]. These created a labeled partition of the object feature space. When presented with the ROI from a new image, their outputs—i.e., the number of each type of object present on an image—are added to the aggregate image data features, and then, image-level classifiers are trained and tested in an n-fold CV regime. Such supervised learning methods are highly useful and may be easily interpreted, but obtaining this information requires significant operator input which may not be available offline or may simply be infeasible online due to the speed of production. When object-level labels are not available, it is necessary to use different approaches to represent the distribution of object features. Such methods do

TABLE V
CLASSIFICATION ACCURACY USING DIFFERENT LEVELS OF USER INPUT

Dataset	CART	C4.5	1NN	9NN	eVQ-Class
1:17 image-level aggregate features only					
Op01 Acc.	91.6	91.8	92.2	90.2	90.5
Op02 Acc.	95.5	95.5	95.1	95.1	95.2
Op03 Acc.	94.2	94.2	93.0	94.3	92.9
Op04 Acc.	94.9	94.1	93.8	92.6	92.9
Op05 Acc.	91.9	90.5	90.4	88.1	85.2
As 1 but with 13 extra features from supervised object classifier					
Op01 Acc.	93.9	93.1	93.4	92.3	92.3
Op02 Acc.	96.7	96.7	96.3	96.2	95.2
Op03 Acc.	96.1	96.9	95.2	94.8	92.6
Op04 Acc.	96.4	95.6	95.4	94.8	92.7
Op05 Acc.	94.2	94.8	93.9	91.8	90.0
As 1 but with 57 extra features -the max. values of the object-level features for each image					
Op01 Acc.	93.2	91.5	92.7	92.4	92.8
Op02 Acc.	95.6	95.1	95.8	95.4	96.1
Op03 Acc.	94.5	94.0	93.4	94.3	93.4
Op04 Acc.	94.9	93.2	94.8	93.5	94.6
Op05 Acc.	92.3	92.1	92.8	91.7	87.1

not require operator input but are highly reliant on the choice of input features. One approach is to apply first-order statistics to this distribution to produce extra image-level descriptors. After some experimentation, it was found that using the maximum value of the object-level features was most useful—e.g., the size/intensity of the biggest/brightest object in an image.

An alternative approach is to apply unsupervised object-level learning to produce a reduced set of extra features which describe the distribution in a different way [6]. In this case, we have proceeded as follows. During the preprocessing stage, a clustering algorithm is applied to all the objects from all the images in the training set. This provides us with C centroids in the object feature space, and also, for each training object, a label from the set $\{1, \ldots, C\}$. Then, for each image in the training set, we can obtain C extra features denoting the number of objects of each type it contains. After this preprocessing stage, the image classifier is then trained using the $17 + C$ features for each labeled image. During testing/validation, each of the segmented ROIs in a new image is assigned to the object class with the nearest centroid, and the appropriate hit-count incremented, to create the $17 + C$ dimensional vector that is input to the classifier. In these experiments, we used a relatively simple clustering algorithm—*k-means*. As this is prone both to settling in local optima and to the effect of redundant features, we applied a Tabu search to perform "wrapper-style" feature selection in the object space (more details can be seen in [49]).

The results of adding different levels of information from the operators are shown in Table V. Table VI shows the effect of augmenting the 17 image-level features with C cluster-based object features and how this changes with C. As can be seen, in each case, the two-level supervised classifier approach improves the results over our baseline. This is not surprising, given that the extra levels of operator input are available, although it is worth noting that training an object classifier is a complex multiclassification approach, where just an accuracy of about 80% can be achieved at the object level. When this level of information was not present, then simply using the first-order statistics to describe the distribution of objects on each image gave significant benefits. This is despite the fact that the result was a 74-D space for the image classifier to learn. As

TABLE VI
CLASSIFICATION ACCURACY (IN PERCENTAGE) WITH 17 IMAGE FEATURES PLUS C CLUSTER-BASED OBJECT FEATURES

Data Set	KM/C4.5/FS (C=4)	KM/C4.5/FS (C=8)	KM/C4.5/FS (C=12)
Op01 Acc.	93.7	94.1	**94.5**
Op02 Acc.	97.1	97.4	**97.4**
Op03 Acc.	95.7	**96.2**	**96.2**
Op04 Acc.	96.2	**96.8**	**96.8**
Op05 Acc.	94.0	93.5	**94.2**

Table VI shows, the results of using the clustering approach to partition the object-level feature space depend on an appropriate choice for C. More importantly, the results demonstrate that, with $C = 12$, the unsupervised cluster-based approach actually does as well, if not better than the supervised approaches. The fact that 12 clusters were found in the data suggests that the problem is one of correctly assigning an object to a class, rather than that those different classes do not actually exist in the data.

VII. ACCOMMODATING PARTIAL CONFIDENCE OF OPERATORS

During the setup phase of an image classification framework, the labeling of several images can be a difficult task for the operators, particularly in cases where real faults are hard to distinguish among themselves or between the so-called pseudoerrors. This problem can become even worse when the operators are not working in the relative calm of an offline setting but are providing real-time decisions at a speed driven by other factors. In this sense, it is promising, sometimes even necessary for the operators to provide information about how confident they are when assigning the labels to certain images or objects. Here, only the confidences in the whole image labels are taken into account. The simplest way is to represent the users' confidence as a value in a range of 0.0 (very unconfident) to 1.0 (very confident). This raises two issues: with what precision should the confidence be used and how should this information be obtained from the users? In keeping with research from other fields about how many categories people can typically discriminate, rather than asking users to spend time thinking of an exact value to assign, we ask them for one of the five values. This means that we can avoid the need to enter those data either via typing or by a mouse click and drag on a slider, both of which are time-consuming operations. The choice of five distinct values, i.e., {20%, 40%, 60%, 80%, 100%} confidence, is also partially driven by the needs of the GUI (see Fig. 2), resulting from an intensive round of discussion and design iteration with industrial users from a range of fields. Based on this, we worked out two principal approaches for incorporating the confidence values for training the classifiers.

The first approach is to apply regression approaches to confidence values transformed by

$$conf\_transformed = \frac{(1 + \eta \cdot conf)}{2} \tag{1}$$

where η takes the value $+1/-1$ for "good"/"bad" samples. Because this model treats the problem as a regression problem rather than a classification one, it is only directly applicable in a two-classification scenario. For multiclass problems, indicator

TABLE VII
CLASSIFICATION ACCURACIES AND IMPROVEMENTS WHEN APPLYING THE REGRESSION APPROACH TO THE 74 AGGREGATED + SUPERVISED FEATURES

Dataset	CART	C4.5	1NN	9NN	eVQ-Class
Op01 Acc.	94.2	-	93.4	92.3	91.9
	(+0.3)	-	(0.0)	(0.0)	(-0.3)
Op02 Acc.	97.2	-	96.3	96.2	96.6
	(+0.5)	-	(0.0)	(0.0)	(+1.4)
Op03 Acc.	97.6	-	95.2	94.8	93.1)
	(+0.7)	-	(0.0)	(0.0)	(+0.5)
Op04 Acc.	96.5	-	95.4	94.8	95.2
	(+0.1)	-	(0.0)	(0.0)	(+2.5)
Op05 Acc.	94.5	-	93.9	91.8	90.6
	(+0.3)	-	(0.0)	(0.0)	(+0.6)

matrices have to be applied (see [20]). Table VII shows the results obtained. We applied the regression variants of CART [5]. This type of regression is not possible with C4.5. For *kNN*, the confidence values of the nearest neighbors of a new incoming instance are averaged, and if being greater than 0.5, label "good" is assigned. For *eVQ-Class*, the hit rate, i.e., the relative proportion of samples, is calculated for each class in the nearest cluster of the new incoming sample by simply summing up all confidence values for each class. The class with the highest sum is assigned to the new incoming instance. The improvement over the results without including the confidence values is clearly visible using CART and *eVQ-Class*. For 1NN and 9NN, there are no improvements. As this method does not provide benefits for several classifiers and does not tackle the multiclass problems, other approaches were investigated.

The second approach tried is based on duplicating the (extended) aggregated feature vectors according to the assigned confidence values. Thus, a feature vector from an image, which is labeled with 1.0 confidence, is duplicated five times, another one labeled with 0.8 confidence is duplicated four times, etc. This means that feature vectors, which are labeled with a higher confidence, are more highly weighted in the training process than those labeled with a lower confidence. In principle, this may not necessarily affect a training process of a classifier; however, the approaches used in this paper (CART, C4.5, kNN, and eVQ-Class) are definitely affected by the density (respectively, relative proportions) of the classes. It should be noted that the idea of creating training sets by sampling from a given data set is not new, e.g., the well-known algorithm bootstrapping [14] does this. However, in one case, the sampling is weighted uniformly, and in the other case, the sampling weights are determined iteratively according to the performance of classifiers built from the initial training sets. What we are proposing is somewhat different, in that the following are true: 1) The weights governing sampling probabilities are a function of the operators' confidence—thus incorporating information which would be lost or ignored by traditional approaches, and 2) the sampling is deterministic, rather than stochastic.

Applying the duplicating feature vector approach on the two-level CD feature data set gives the results shown in Table VIII. As can be seen, the results improve for all classifiers except 1NN and can outperform the regression approach significantly (see Table VII). It is to be expected that no difference is observed with 1NN, as, of course, duplication has no effect when only one instance is considered to make each decision. In contrast, when larger groups of neighbors are used (9NN—column 5), the increase can be dramatic as "confident" images outvote others. Not only does this technique give improvements for all the different types of classifiers but also it does so for all operators: toward 98% for operators #2 and #3 and toward 97% for operator #4.

TABLE VIII
CLASSIFICATION ACCURACY USING TWO-LEVEL APPROACH AND DUPLICATING FEATURE VECTORS ACCORDING TO CONFIDENCE LEVELS OF OPERATORS. RESULTS IN BRACKETS SHOW THE IMPROVEMENT IN PERCENTAGE BY EACH CLASSIFIER OVER TWO-LEVEL APPROACH WITHOUT DUPLICATION

Dataset	CART	C4.5	1NN	9NN	eVQ-Class
Op01 Acc.	94.4	94.0	93.4	94.2	93.4
	(+0.5)	(+0.3)	(+0.0)	(+1.9)	(+1.2)
Op02 Acc.	97.7	96.9	96.3)	96.2	96.7
	(+1.0)	(+0.8)	(+0.0)	(+0.0)	(+1.5)
Op03 Acc.	98.0	97.3	95.2	95.2	96.5
	(+1.1)	(+0.1)	(+0.0)	(+0.4)	(+3.9)
Op04 Acc.	97.0	96.7	95.4	95.9	95.6
	(+0.6)	(+1.1)	(+0.0)	(+1.1)	(+2.9)
Op05 Acc.	95.2	95.3	93.9	93.0	91.7
	(+1.0)	(+0.5)	(+0.0)	(+1.2)	(+1.7)

VIII. PROVIDING INSIGHT INTO CLASSIFIER STRUCTURES

The operator may wish to have more insight into the classifier structures in order to obtain a better understanding of the classifier decisions as well as the characteristics of the faulty situations at the systems. Sometimes, he may even wish to interact with the classifier's training process, i.e., to modify some structural components of the classifier (e.g., rules, leafs, neurons, etc.). This is because he has some intrinsic knowledge about the fault and nonfault characteristics in the recorded images. Note that this will lead finally to a kind of gray-box modeling behavior [34] in an alternating scheme, which is to train an initial classifier based on some training data, then to modify the trained classifier by an expert knowledge, and then to adapt further the classifier with new online data. In order to gain more insight into the classifier structure, a first issue is to consider which model architecture is a reliable one to do so, i.e., yielding transparent classifiers. From the repertoire of methods that were used in this paper, the decision-tree approaches seem to be most convenient, as they produce a classifier with a tree structure, where each path from the root to the terminal node can be transferred to a linguistically readable classification rule [32], [37].

A second important issue is to consider complexity reduction processes for making trained complex models slimmer and hence more transparent and interpretable. For decision trees, this is achieved by so-called pruning techniques, which perform complexity reduction based on nested tree sequences (from simple to more complex trees) [33]. For an example of a simple (pruned) tree structure, see the right image in Fig. 5, whereas the left image represents the unpruned (quite unclear) tree. For the pruned tree, the classification rules can be read as follows (xi denotes the ith feature).

IF $x7 \geq 4.72$ THEN Image is "BAD."
IF $x7 \leq 4.72$ AND $x12 < 3384.5$ THEN Image is "GOOD."
IF $x7 \leq 4.72$ AND $x12 \geq 3384.5$ AND $x2 \geq 3.38$ THEN Image is "GOOD" and so on.

In fact, this means that if feature $x7$ is greater or equal than 4.72, the image is already classified as "BAD." In order to

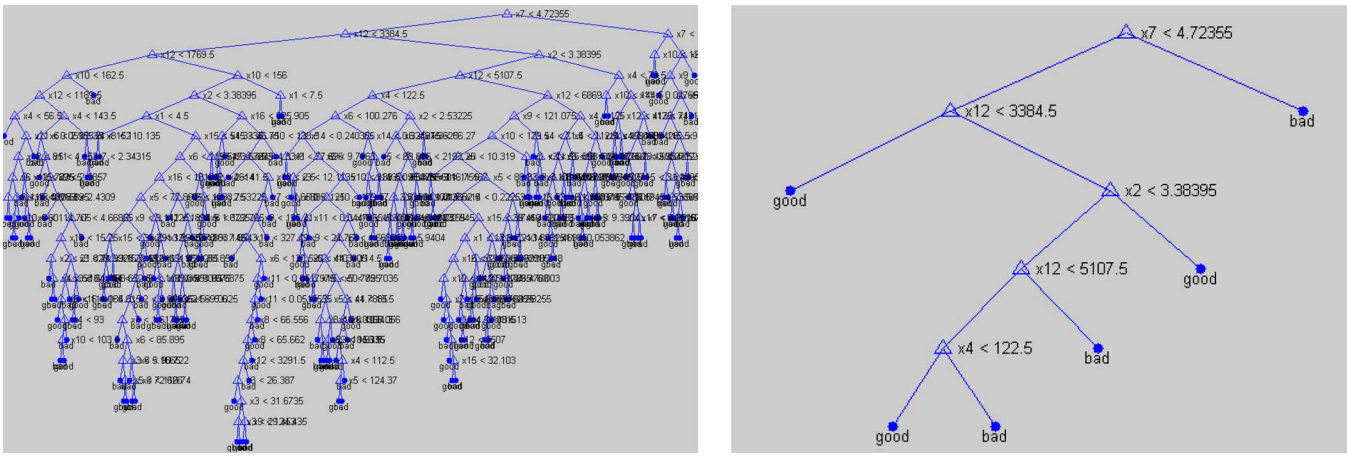

Fig. 5. (Left) Complex decision tree not being pruned. (Right) By applying a pruning step, the tree gets much easier to interpret by loosing only a fraction of its predictive power.

give these rules a better linguistic representation, they can be cross-checked what 4.72 means with respect to the range of the feature $x7$, which is, in this case, the aggregated feature "maximal intensity of an object," whereas intensity is defined by multiplying the size with the average (normalized) gray values of the object: 4.72 is a quite low value when taking into account that this feature ranges from 0.078 to 98.14; this means that the rule "IF $x7 \geq 4.72$ THEN Image is BAD" can be transferred to the linguistic rule "IF the maximal intensity of an object is NOT LOW THEN Image is BAD" as well as "IF there are any ROIs in the (contrast) image which are clearly visible THEN image is BAD." In this sense, the user can decide whether it is a correct or wrong rule.

Please note that pruning is not only for obtaining a better transparency of the classification rules and more compact representation of the intrinsic knowledge in the training data but also for improving the accuracies of the classifiers, as complex trees tend to overfit on new unseen samples. In this sense, pruning is definitively always a successful step when designing a decision-tree-based classifier. The results in Table VIII with respect to the CART algorithm were also achieved with an implicit pruning step, with the pruning level set to 1 in this case, leading to a still quite complex tree with a depth of 11 and number of nonterminal nodes of 37. It was also examined how a stronger pruning improves the transparency (measured in terms of two numbers: tree depth and number of nodes) while weakening the predictive powers. This is underlined in Fig. 6, where, for the training data labeled by Operator 6, the number of nonterminal nodes versus the elicited accuracy is visualized. Here, it can be seen that, indeed, by using a pruning level of 1, the best accuracy can be obtained; however, when increasing the pruning level to a value of 6 and decreasing the number of nonterminal nodes (i.e., the number of rules) to just 9, only a fraction (i.e., 0.8%) of the accuracy is lost. In this sense, this pruned tree represents a good tradeoff between performance in terms of accuracy and transparency of the classifier.

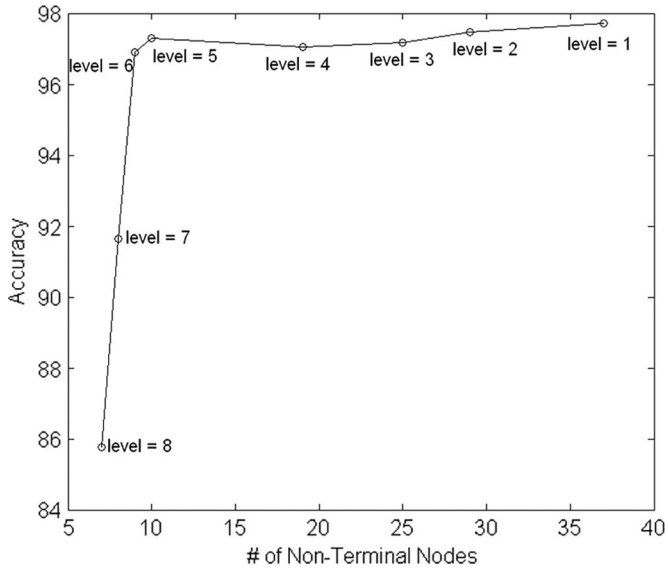

Fig. 6. Complexity versus accuracy of the pruned decision tree.

IX. Conclusion and Outlook

As ML systems move out of the laboratory and into real-world applications such as vision and image processing, it is valuable to reconsider some of the assumptions that have been made about how such systems can best learn from users. In this paper, we have discussed some important issues and suggested how they can be handled. Experiments conducted with "real" data within the context of a generic image processing system show that, when properly handled, the human factors can represent an additional form of information to these systems for improving performance and may widen the applicability and usability, rather than to be a disagreeable source of noise. The key issues of these factors include online guidance and feedback, a diversity of user skills, and uncertainties as well as different levels of knowhow and detail in users' input. Crucial to this change is paying a proper consideration to the way that the users actually interact with the system. They can be distilled to two questions. The first of these is as follows: What should the users be shown and how? Many years of experience of the project partners have taught us that maintaining users' involvement in the value of the system is crucial, so particularly during online training, it is vital for the interface to show that the system is actually learning from their input. The second question is as follows: What information should the users

provide, and how? We have shown how asking for too much information—e.g., object-level labels—can sometimes cause complications and may not always be possible. In contrast, other information such as the human's uncertainty can be easily obtained and lead to significant performance improvements. The improvements are made possible by recent advances in the speed with which GUIs can operate. The next generation of user interaction devices offers the potential to build on this research, creating much richer human–machine learning interaction.

ACKNOWLEDGMENT

This paper reflects only the authors' views.

REFERENCES

[1] E. J. Bass and A. R. Pritchett, "Human-automated judge learning: A methodology for examining human interaction with information analysis automation," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 4, pp. 759–776, Jul. 2008.

[2] H. Bekel, I. Bax, G. Heidemann, and H. Ritter, "Adaptive computer vision: Online learning for object recognition," in *Proc. 26th DAGM Symp. Pattern Recog.*, Tübingen, Germany, 2004, pp. 447–454.

[3] P. Bottoni, S. K. Chang, M. F. Costabile, S. Levialdi, and P. Mussion, "Modeling visual interactive systems through dynamic visual languages," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 32, no. 6, pp. 654–669, Nov. 2002.

[4] A. Bouridane, M. A. Tahir, and F. Kurugollu, "Simultaneous feature selection and feature weighting using hybrid tabu search/k-nearest neighbor classifier," *Pattern Recognit. Lett.*, vol. 28, no. 4, pp. 438–446, Mar. 2007.

[5] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Boca Raton, FL: Chapman & Hall, 1993.

[6] P. Caleb-Solly and J. E. Smith, "Adaptive surface inspection via interactive evolution," *Image Vis. Comput.*, vol. 25, no. 7, pp. 1058–1072, Jul. 2007.

[7] P. Caleb-Solly and M. Steuer, "Classification of surface defects on hot rolled steel using adaptive learning methods," in *Proc. IEEE 4th Int. Conf. Knowl.-Based Intell. Eng. Syst. Allied Technol.*, Brighton, U.K., 2000, vol. 1, pp. 103–108.

[8] G. A. Carpenter and S. Grossberg, "Adaptive resonance theory (art)," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 79–82.

[9] E. Castillo and E. Alvarez, *Expert Systems: Uncertainty and Learning*. New York: Springer-Verlag, 2007.

[10] S. Cho and J. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 2, pp. 380–384, Feb. 1995.

[11] R. Cipolla and A. Pentland, Eds., *Computer Vision for Human-Machine Interaction*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[12] M. F. Costabile, D. Fogli, P. Mussion, and A. Piccinno, "Visual interactive systems for end-user development: A model-based design methodology," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 6, pp. 1029–1046, Nov. 2007.

[13] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Chichester, U.K.: Wiley-Interscience, 2000.

[14] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. London, U.K.: Chapman & Hall, 1993.

[15] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. KDD*, Portland, OR, 1996, pp. 226–231.

[16] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[17] J. Fürnkranz and A. K. Seewald, "An evaluation of grading classifiers," in *Proc. 4th Int. Conf. Adv. Intell. Data Anal.*, Cascais, Portugal, 2001, pp. 115–124.

[18] F. Gayubo, J. L. Gonzalez, E. De La Fuente, F. Miguel, and J. R. Peran, "On-line machine vision system for detect split defects in sheet-metal forming processes," in *Proc. 18th Int. Conf. Pattern Recog.*, Los Alamitos, CA, 2006, pp. 723–726.

[19] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, no. 2, pp. 4–29, Apr. 1984.

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. New York: Springer-Verlag, 2001.

[21] A. Jaimes and N. Sebe, "Multimodal human-computer interaction: A survey," *Comput. Vis. Image Underst.*, vol. 108, no. 1/2, pp. 116–134, Oct. 2007.

[22] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[23] N. Kasabov, *Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines*. London, U.K.: Springer-Verlag, 2002.

[24] J. Kittler, M. Hatef, R. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, Mar. 1998.

[25] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. New York: Wiley, 2004.

[26] L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin, "Decision templates for multiple classifier fusion: An experimental comparison," *Pattern Recognit.*, vol. 34, no. 2, pp. 299–314, 2001.

[27] L. I. Kuncheva, C. J. Whitaker, C. A. Shipp, and R. P. W. Duin, "Limits on the majority vote accuracy in classifier fusion," *Pattern Anal. Appl.*, vol. 6, no. 1, pp. 22–31, Apr. 2003.

[28] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, NJ: Prentice-Hall, 1999.

[29] E. Lughofer, "Extensions of vector quantization for incremental clustering," *Pattern Recognit.*, vol. 41, no. 3, pp. 995–1011, Mar. 2008.

[30] E. Lughofer, "FLEXFIS: A robust incremental learning approach for evolving TS fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1393–1410, Dec. 2008.

[31] E. Lughofer, P. Angelov, and X. Zhou, "Evolving single- and multi-model fuzzy classifiers with FLEXFIS-Class," in *Proc. FUZZ-IEEE*, London, U.K., 2007, pp. 363–368.

[32] S. Mitra, K. M. Konwar, and S. K. Pal, "Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: Generation and evaluation," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 4, pp. 328–339, Nov. 2002.

[33] K. Muata and O. Bryson, "Post-pruning in decision tree induction using multiple performance measures," *Comput. Oper. Res.*, vol. 34, no. 11, pp. 3331–3345, Nov. 2007.

[34] O. Nelles, *Nonlinear System Identification*. Berlin, Germany: Springer-Verlag, 2001.

[35] G. Papari and N. Petkov, "Algorithm that mimics human perceptual grouping of dot patterns," in *Brain, Vision, and Artificial Intelligence*. Berlin, Germany: Springer-Verlag, 2005, pp. 497–506.

[36] S. J. Qin, W. Li, and H. H. Yue, "Recursive PCA for adaptive process monitoring," *J. Process Control*, vol. 10, no. 5, pp. 471–486, 2000.

[37] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[38] S. Raiser, E. Lughofer, C. Eitzinger, and J. E. Smith, "Impact of object extraction methods on classification performance in surface inspection systems," *Mach. Vis. Appl.*, 2009. DOI: 10.1007/s00138-009-0205-z, to be published.

[39] J. Rasmussen, *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*. New York: Elsevier, 1986.

[40] G. Rogova, "Combining the results of several neural network classifiers," *Neural Netw.*, vol. 7, no. 5, pp. 777–781, 1994.

[41] D. Sannen, H. Van Brussel, and M. Nuttin, "Classifier fusion using discounted Dempster-Shafer combination," in *Proc. Int. Conf. MLDM*, Leipzig, Germany, 2007, pp. 216–230.

[42] D. Sannen, M. Nuttin, J. E. Smith, M. A. Tahir, E. Lughofer, and C. Eitzinger, "An interactive self-adaptive on-line image classification framework," in *Proc. ICVS*, vol. 5008, *Lecture Notes in Computer Science*, A. Gasteratos, M. Vincze, and J. K. Tsotsos, Eds., Berlin, Germany, 2008 pp. 173–180.

[43] G. H. Shakouri and M. B. Menhaj, "A systematic fuzzy decision-making process to choose the best model among a set of competing models," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 5, pp. 1118–1128, Sep. 2008.

[44] A. Shilton, M. Palaniswami, D. Ralph-D, and A. C. Tsoi, "Incremental training of support vector machines," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 114–131, Jan. 2005.

[45] J. E. Smith and M. A. Tahir, "Stop wasting time: On predicting the success or failure of learning for industrial applications," in *Proc. 8th Int. Conf. IDEAL*, vol. 4881, *Lecture Notes in Computer Science*, Berlin, Germany, 2007, pp. 673–683.

[46] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *J. R. Stat. Soc.*, vol. 36, no. 1, pp. 111–147, 1974.
[47] A. G. Sutcliffe and A. Gregoriades, "Automating scenario analysis of human and system reliability," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 2, pp. 249–261, Mar. 2007.
[48] M. A. Tahir and J. Smith, "Improving nearest neighbor classifier using tabu search and ensemble distance metrics," in *Proc. IEEE ICDM*, Hong Kong, 2006, pp. 1086–1090.
[49] M. A. Tahir, J. E. Smith, and P. Caleb-Solly, "A novel feature selection based semi-supervised method for image classification," in *Proc. ICVS*, vol. 5008, *Lecture Notes in Computer Science*, A. Gasteratos, M. Vincze, and J. K. Tsotsos, Eds., Berlin, Germany, 2008, pp. 484–493.
[50] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 1, pp. 116–132, Feb. 1985.
[51] S. Tong and D. Keller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, no. 1, pp. 45–66, Mar. 2002.
[52] G. Valentini and F. Masulli, "Ensembles of learning machines," in *Neural Nets WIRN Vietri-02*, M. Marinaro and R. Tagliaferri, Eds. Heidelberg, Germany: Springer-Verlag, 2002, ser. Series Lecture Notes in Computer Sciences.
[53] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, Oct. 1996.
[54] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 405–410, Apr. 1997.
[55] X. Wu, V. Kumar, J. R. Quinlan, J. Gosh, Q. Yang, H. Motoda, G. J. MacLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, Dec. 2007.

Edwin Lughofer received the Ph.D. degree from Johannes Kepler University of Linz, Linz, Austria.

He is currently a Postdoctoral and Research Assistant with Johannes Kepler University of Linz. During the last seven years, he participated in several international research projects such as the EU Project DynaVis (http://www.dynavis.org), EU Project AMPA, or EU Project Syntex (http://www.syntex.or.at). In this period, he has published around 40 journal and conference papers in the fields of evolving fuzzy systems, machine learning and vision, clustering, fault detection, image processing, and human–machine interaction. In these research fields, he acts as a reviewer in peer-reviewed international journals and a (co-)organizer of special sessions. Research visits and stays include the following locations: Center for Bioimage Informatics and the Department of Biological Sciences, Carnegie Mellon University, Pittsburgh, PA; Faculty for Informatics (ITI), Otto-von-Guericke-Universität, Magdeburg, Germany; and the Department of Communications Systems InfoLab21, Lancaster University, Lancaster, U.K.

Dr. Lughofer was a recipient of the Best Paper Award at the International Symposium on Evolving Fuzzy Systems in 2006, the Royal Society Grant for know-how exchange with Lancaster University in the field of evolving fuzzy systems in 2007, and an award at the Third Genetic and Evolving Fuzzy Systems Workshop in 2008.

James E. Smith received the B.S. degree in electrical sciences from the University of Cambridge, Cambridge, U.K., and the Ph.D. degree in 1998.

He worked in the industry for several years before returning to academia. He is currently a Reader in artificial intelligence with the University of the West of England, Bristol, U.K. His current research interests include the theory and application of intelligent systems that adapt their learning strategies in response to experience and the interface between humans and adaptive intelligent systems. He has researched and published extensively in the field of evolutionary computation.

Muhammad Atif Tahir received the B.E. degree in computer systems engineering from NED University of Engineering and Technology, Karachi, Pakistan, the M.Sc. degree in computer engineering from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, and the Ph.D. degree from Queen's University, Belfast, U.K.

He was a Research Fellow with the Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England, Bristol, U.K. He is currently working as a Research Fellow with the Centre for Vision, Speech and Signal Processing, University of Surrey, Surrey, U.K. His current research activities include pattern recognition, field programmable gate arrays, machine learning, digital signal processing, image processing, evolutionary heuristics, and computer networks.

Praminda Caleb-Solly received the M.Sc. degree in biomedical instrumentation engineering.

She trained as an Electronic Engineer. She is currently a Senior Lecturer with the Bristol Institute of Technology, University of the West of England, Bristol, U.K. Her teaching specialism is human–computer interaction. Her current research focus in interactive evolutionary computation is on enhancing user experience and ease of interaction, which are vital to ensure the applicability of these algorithms in real-world scenarios. Her research also includes the application of adaptive nonlinear learning algorithms to a range of medically related problems.

Christian Eitzinger received the B.S. degree in mechatronics from Johannes Kepler University of Linz, Linz, Austria, the M.Sc. degree in 1995, and the Ph.D. degree in training algorithms for neural networks in 2001.

He is currently the head of the Department of Machine Vision, Profactor GmbH, Steyr, Austria, where he supervises and coordinates eight research employees. During the last few years, he coordinated three EU projects and several national research projects. His research fields include image processing, machine vision and learning, human–machine interaction, and neural networks, where he published around 30 papers in peer-reviewed international journals and conferences.

Dr. Eitzinger was a recipient of the Erwin-Wenzel award in 2001.

Davy Sannen received the M.Sc. degree in computer science in 2004 from the Transnationale Universiteit Limburg, Diepenbeek, Belgium, and the Master of Artificial Intelligence degree in 2005 from the Katholieke Universiteit Leuven, Leuven, Belgium, where he is currently working toward the Ph.D. degree in mechanical engineering.

His research focuses on pattern recognition and classifier combination for data mining and quality control.

Marnix Nuttin received the M.Sc. degree in electrical engineering and the Ph.D. degree in mechanical engineering from the Katholieke Universiteit Leuven (K.U.Leuven), Leuven, Belgium, in 1992 and 1998, respectively.

He is currently with the Faculty of Engineering, K.U.Leuven. His main research interests include learning approaches to robotics and manufacturing, including learning techniques for programming by demonstration and adaptive shared autonomy for wheelchair control.