Inter-Active Learning of Ad-Hoc Classifiers for Video Visual Analytics

Benjamin Höferlin*[a], Rudolf Netzel†[b], Markus Höferlin[b], Daniel Weiskopf[b], Gunther Heidemann[a]

[a]Institute of Cognitive Science, University of Osnabrück
[b]Visualization Research Center (VISUS), University of Stuttgart

ABSTRACT

Learning of classifiers to be used as filters within the analytical reasoning process leads to new and aggravates existing challenges. Such classifiers are typically trained ad-hoc, with tight time constraints that affect the amount and the quality of annotation data and, thus, also the users' trust in the classifier trained. We approach the challenges of ad-hoc training by *inter-active* learning, which extends active learning by integrating human experts' background knowledge to greater extent. In contrast to active learning, not only does inter-active learning include the users' expertise by posing queries of data instances for labeling, but it also supports the users in comprehending the classifier model by visualization. Besides the annotation of manually or automatically selected data instances, users are empowered to directly adjust complex classifier models. Therefore, our model visualization facilitates the detection and correction of inconsistencies between the classifier model trained by examples and the user's mental model of the class definition. Visual feedback of the training process helps the users assess the performance of the classifier and, thus, build up trust in the filter created. We demonstrate the capabilities of inter-active learning in the domain of video visual analytics and compare its performance with the results of random sampling and uncertainty sampling of training sets.

Index Terms: H.3.3 [Information Systems]: Information Storage and Retrieval—Information Search and Retrieval; I.2.6 [Computing Methodologies]: Artificial Intelligence—Learning

1 INTRODUCTION

Reduction of data to its relevant parts is a central and recurrent step of the visual analytics process, as outlined in the visual analytics mantra [17]: "Analyse First – Show the Important – Zoom, Filter and Analyse Further – Details on Demand". Such reduction is critical for data scalability and is performed by automatic methods or user-defined filters. While automatic methods reduce the amount of data by exploiting some structure or by calculating predefined features and statistics, filters serve as their equivalents in human visual information seeking. Filters are involved in both exploratory interaction to reduce the amount of data displayed and to focus on the details (e.g., by dynamic queries [29]) and in confirmatory interaction to confirm or refute hypotheses about the data (e.g., in video visual analytics [15]). Typically, users define filters by providing model parameters or examples of data instances they want to be included in, or excluded from, their query.

We focus on the question how filters can be efficiently defined. This question arises especially when analyzing complex and high-dimensional data spaces, where appropriate model parameters are unknown and filter definition by a single example is too weak. In such cases—we will use the examples of data and tasks from video visual analytics throughout the paper—query by multiple examples can be useful, which is identical to the training of a complex classifier. In this way, users can specify what they seek by integrating machine learning techniques into information visualization, as commonly recommended (e.g., by Shneiderman [30] or Chen [6]). However, in contrast to pre-trained classifiers as they are widely used in video analytics (e.g., person or car detectors, included in many video management systems), classifiers used to define filters within the visual analytics process have to be trained ad-hoc.

Such ad-hoc trained classifiers for filtering are required within the sense-making loop of analysts [35], when they build a case or search for support or evidence for a hypothesis within the data. Let us consider the example of video surveillance operators who assume, after some initial analysis of video sequences, that a cyclist might have been involved in the case of a traffic incident they deal with. Hence, they want to extract cyclists from video data to reduce the amount of video and to focus on promising parts for hypothesis verification. This example illustrates the need for training of new and arbitrary classifiers that can also be highly complex and specialized (e.g., hand-waving bicyclists with red helmets may be important in our example scenario). Since pre-trained instances of such classifiers are generally not available, the analysts have to define the filter by themselves. However, feature selection and model parameter definition for objects such as a bicyclist are too complex to be manually defined, even for domain experts with support by interactive visualization [43]. Hence, filter definition via query by examples promises to be the only viable solution.

In contrast to traditional supervised training of a classifier, ad-hoc training involves new challenges:

Annotation Costs: Data annotation is a very costly task because a large amount of annotated data is required for proper training of a classifier. Furthermore, the data has often to be annotated by domain experts in a time-consuming process. These facts question the benefit of ad-hoc training of filters within the analytical reasoning process. In addition, the issue of decreasing analysis performance arises if disruption that comes with high time consumption for training influences the analysis process. Finally, to find appropriate examples that can be labeled tends to be a difficult task that becomes worse with the rareness of the data instances queried (imagine the search for hand-waiving cyclists with red helmets).

Annotation Quality: A typical phenomenon of positive example selection in ad-hoc training scenarios is that the provided data instances are not sampled as independent and identically distributed random variables from the query distribution, as required by the learner. In fact, users tend to provide samples drawn from a rather small region in data space. This effect obviously increases with the rareness of suitable examples in the data: if a hand-waiving cyclist was found, he will be annotated in all frames of the video. Although this approach provides multiple training examples, the training set itself is very specialized and may not generalize to all instances of the intended query (e.g., all hand-waiving cyclists). Furthermore, the quality of annotated examples is affected by the often vague idea of the query the users have in mind when defining the filter. The question of the exact range of the data instances of interest may arise (e.g., is a person on a trike also of interest?). This issue is further intensified in data domains where no clear data instances are

*e-mail:benjamin.hoeferlin@uni-osnabrueck.de
†e-mail:rudolf.netzel@visus.uni-stuttgart.de

IEEE Conference on Visual Analytics Science and Technology 2012
October 14 - 19, Seattle, WA, USA
978-1-4673-4753-2/12/$31.00 ©2012 IEEE

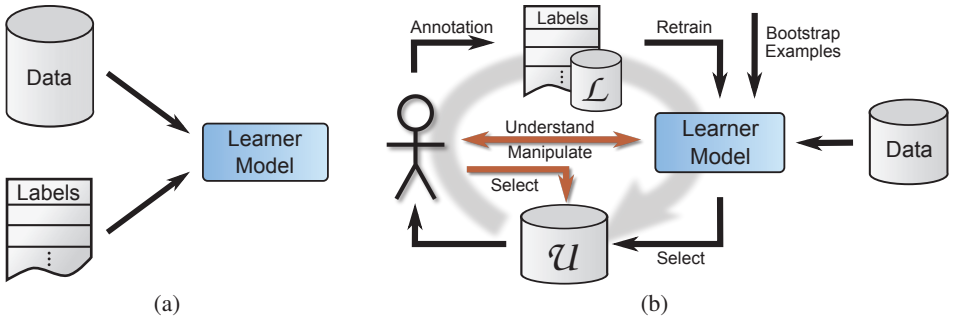

Figure 1: (a) Data flow of passive supervised machine learning. A training set (data + large amount of labels) is provided to the learner, either as a monolithic set (batch learning) or in many smaller pieces (online learning). (b) In active learning, a bootstrapped learner iteratively refines itself by posing queries from a pool of unlabeled data $\mathcal{U}$ to a (human) oracle that provides labels for the data $\mathcal{L}$. Inter-active learning is an extension (red arrows) of active learning that further allows the human experts to directly integrate their background knowledge into the model.

predefined. For example, we encounter the question how a data instance should be defined in object or event detection for video analysis: is the object silhouette the right way to crop the example from the video frame or can we use a bounding rectangle to define the image region, and if so, does the precision of the rectangle affect the filter performance? Such questions will not arise in domains where precisely defined documents are available, as it is the case in text document retrieval or content-based image retrieval. These issues especially arise in the ad-hoc training context. We will refer to them by the term annotation noise (see Fig. 2).

Classifier Quality Assessment: A general question in machine learning is to detect the appropriate moment when to stop training of a classifier. The classifier should well adapt to the training data, but be general enough to correctly classify unseen data instances, too. This overfitting issue is traditionally tackled by cross validation. However, this approach requires either much more annotated examples for a validation set or much more time, since multiple classifiers have to be trained on different subsets of the training set. Furthermore, the generalization assessment of the classifier by cross validation is limited by the sampling bias induced by ad-hoc annotation (cf. annotation quality). Thus, cross validation is hardly feasible and often undesired in the ad-hoc training context. A question related to the stopping criteria in training is the question of stopping criteria in annotation of examples. This issue is of practical relevance because annotation involves costs. However, often no appropriate measure of training progress is available that facilitates the definition of such stopping criteria [36]. Finally, quality assessment of the classifier becomes important to ad-hoc training because the users have to develop trust in their defined and applied filters. Hence, it is important to them to judge their classifier's performance when it is faced with unseen data or noisy data.

Training of a classifier under the constraint of high annotation cost is tackled by the field of *active learning*. In contrast to traditional supervised machine learning (see Fig. 1 (a)), an active learner is allowed to choose the data from which it wants to learn. This way, typically greater accuracy can be achieved with fewer training labels. As depicted in Fig. 1 (b), active learning is an iterative process of refinement in which the learner may pose queries of unlabeled data to an oracle (e.g., a human) that provides the labels for this data. The active learner typically queries labels for the data instances with the highest informativeness or those that promise to reduce uncertainty most. Settles [27] and Olsson [21] provide an introduction and comprehensive overview of the field of active learning. For a survey of the application of active learning for multimedia annotation, we refer to Wang and Hua [41].

Theoretical analysis has shown that an active learner (however, a computationally complex query-by-committee approach) can reduce the complexity of required labels in exponential order [10]. Empirical analysis reveals that in the majority of applications, active learning is able to reduce the amount of labels [27], too. A survey of the usage of active learning for text annotation exhibits that the expectations of most practitioners on the performance of active learning are either fully (36.3%) or partially (54.4%) met [36]. The reduction of examples to label in practical situations, however, may lie far behind exponential decrease and, dependent on the dataset, sometimes does not show any improvement at all [24]. Further, some authors report negative results [11, 13] or show that the performance depends on the expertise of the annotator [2].

Due to these results and the further challenges that we face in the context of ad-hoc training, we question whether active learning alone is suitable for ad-hoc training and capable of meeting the tight time constraints existing in this application. Furthermore, we question the idea of the human experts as mere annotators, but believe that their expertise should be utilized in a more direct way.

In this paper, we introduce a novel method called *inter-active learning*, an extension to conventional active learning that directly involves human experts in the ad-hoc training process using the visual analytics methodology. The additional interaction introduced by inter-active learning is depicted by red arrows in Fig. 1 (b): the goal of this process is to efficiently create filters leveraging the complementary strengths of human and machine, as outlined by Bertini and Lalanne [4] in the context of the knowledge discovery process.

In detail, inter-active learning efficiently approaches the goal of a well-trained classifier by iterating over the three basic steps: i) assessment of the performance of the classifier, ii) annotation of data instances and/or manipulation of the classifier model, and iii) retraining of the classifier. In Section 3, we will break down these basic steps into different tasks the users perform in each cycle to incorporate new background knowledge into the trained model.

This paper contributes to the current state of research by presenting a way how the problem of ad-hoc training of classifiers for interactive filter definition can be tackled by visual analytics. Besides introducing a general methodology, which can be considered as an extension of active learning methods toward increasing leverage of the users' expertise, we apply inter-active learning to the domain of video analysis for validation. We present an integrated visual analytics system that covers the three steps of inter-active learning and provides an adaption to online learning of the cascade classifier model we use, as well as visualization and interaction models that can cope with the complexity, high dimensionality, and huge amount of data of the video domain. A usage scenario illustrates the different tasks the users carry out within the three steps and further provides validation of our method by comparison to classifier training with random sampling and uncertainty sampling.

2 Related Work

Our approach is related to techniques from visual analytics, knowledge discovery, data mining, information visualization, and machine learning. Closely related is previous work by Seifert and Granitzer [25], Seifert *et al.* [26], May and Kohlhammer [18], and Heimerl *et al.* [14]. All four methods aim at tight integration of the user into the labeling process. The first work [25] presents a user-based and visually supported active learning method that was successfully validated on different multi-class datasets of various data domains. Seifert *et al.* [26] focus on visual classifier performance assessment, while the learner can be adapted by labeling data instances in an information landscape visualization. Validation is provided by means of a multi-class, multi-label text classification scenario. May and Kohlhammer [18] also allow the user to refine a classifier model by selection of training examples. They focus on performance feedback of the classifier model using a visualization that facilitates pre-attentive pattern identification. Recently, Heimerl *et al.* [14] have introduced a system to interactively train a support vector machine model for text document retrieval based on visual analytics and active learning. Further, they provided a thorough evaluation of their approach. However, all four methods do not go beyond interactive definition of the training set and user selection of the data instances to label. More direct integration of human experts' background knowledge is not considered.

There are several approaches of interactive definition of classifiers (e.g., [1, 8, 32, 34, 43]), that do not consider support of (semi-) automatic methods for classifier refinement, such as active learning. Although these methods were successfully applied to train or combine classifiers, purely user-based definition of classifiers seems only to be viable for problems with low complexity [43]; otherwise, they annoy the users by recurring tasks [32]. Furthermore, interactive definition often constrains the complexity of the classifier model; hence, decision trees typically appear in these works.

Another approach incorporating the users' background knowledge into the training process is followed by the active learning community. In the domain of text classification, where features often coincide with words, promising methods were developed that allow the learner to pose queries to the oracle to label features, instead of just data instances [28, 31]. This means that the oracle can tell the learner if a particular feature describes a particular class well. However, feature labeling can only be applied in areas where features are tangible to human users (e.g., word features in text classification). Thus, such methods are not applicable in complex and abstract problem environments in which features do not exhibit any concrete symbolic meaning to humans. Besides the methods that incorporate human decisions in machine learning, visualizations of classifier models for performance assessment and model understanding, such as [3, 5, 19], are naturally related to our approach. In contrast to the existing approaches, we advance the fields of active learning and visual classifier definition by combining them to a visual analytics process called inter-active learning.

3 Inter-Active Learning

In this section, we outline the theoretical considerations of inter-active learning and specify its requirements for learning models, visualization, and interaction.

Inter-active learning re-formulates the problem of supervised machine learning as a visual analytics problem. Supervised machine learning for classification, as depicted in Fig. 1 (a), seeks to find model parameters m that minimize the class confusion error E (according to the distance function f_{error}) of the classifier function $h_m : S \to T$, which maps the data distribution S to a set of target classes $T = \{0, 1, \cdots, n\}$:

$$E = \sum_{i \in I} f_{\text{error}}(h_m(s_i), l_i)$$

Here, a training vector of data/label pairs (s_i, l_i) is provided, where data samples s_i are drawn as independent and identically distributed random variables from S and data labels $l_i = \mathscr{L}(s_i)$ are given by the labeling function $\mathscr{L}$; I denotes the set of sample indicies. We do not consider any further regularization terms, such as smoothness of the function.

In contrast to passive supervised learning, active learning additionally minimizes the costs $C = \sum_{i \in I} \mathscr{C}(\mathscr{L}(s_i))$ that arise by acquiring a finite set of labels l_i. Active learners are allowed to pose query of data instances to be labeled to the users. The decision which data has to be labeled for the next training cycle depends on the current model m, the available data instances $s_i \in \mathcal{U}$, and the labeling costs. Active learners typically assume a uniform cost function $\mathscr{C}$ (i.e., $C = |I|$) and thus query labels for the most informative data instances from the users. This process is illustrated for a pool-based active learner in Fig. 1 (b) (ignoring the red arrows).

Inter-active learning, as an extension to active learning, pursues the same objective: a well-trained classifier trained with minimal labeling costs. In contrast to active learning, inter-active learning assumes that the users—based on their expertise of the domain and their knowledge about m—are able to select a more effective set of data instances to be labeled. Further, the training process can benefit from direct modifications of the learner model m. In high-dimensional data domains with complex dependencies, however, direct definition of model parameters can be difficult [43]. Hence, we focus on permitting the users to detect and solve contradictions between their domain knowledge and the actually trained model m.

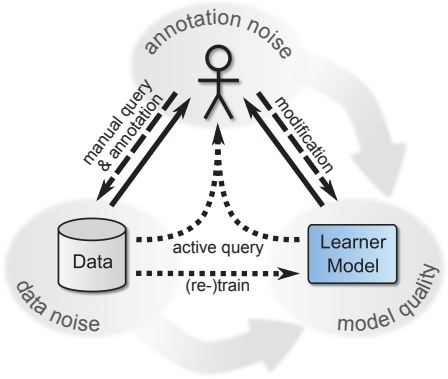

Figure 2: Major components and information flow involved in the inter-active learning process. Solid lines depict information conveyed to the users by visualization, dashed lines represent user interaction, and dotted lines illustrate the flow of information triggered by automatic methods. Furthermore, the contribution of noise-affected data and labels to the model's uncertainty is depicted.

Due to the tight connection between learner and user, visualization and human-computer interaction are, besides automatic methods, the central aspects of inter-active learning. Figure 2 illustrates this connection between the three major elements: learner model, user, and data. Furthermore, Fig. 2 depicts the flow of information between the three elements that can be assigned to one of the three category of methods: visualization, interaction, or automatic method. The iterative interaction of these three components results in a visual analytics process that aims to refine the classifier model and its comprehension by the users.

Each cycle of this iterative process consists of three steps mentioned before: i) assessment of the model performance, ii) refinement of the classifier model, and iii) retraining of the classifier. The first two steps, which include user interaction, can further be divided into tasks the users may consider to process each cycle. For the first step, these tasks include assessing the success of the last

training cycle (training feedback) and determining if a stopping criterion was reached (e.g., the model has already reached an appropriate level of quality or training does not improve the model anymore). Furthermore, by assessing the model's performance, users can build trust in their trained model and learn to know its strengths and weaknesses. Hence, they can incorporate the performance and uncertainty of "their" filters into their decisions within the analytical reasoning process. Finally, quality assessment also guides the users in refining the model. Users may detect overfitting of the model, low robustness to noise, or lack of generalization. These issues are tackled in the second step of a cycle.

After the classifier model was analyzed in the first step, two types of refinement are available to the users in the second step: data annotation and model manipulation. While both data annotation and model manipulation can be used to broaden the classifier model to accept a wider variety of data instances or to narrow the acceptance range, we recommend using model manipulation mainly for generalization purposes; in contrast, data annotation is suitable for both tasks. This recommendation accounts for the complex dependencies of high-dimensional data distributions. In such cases, it is often easier to tell the system what is wrong (e.g., overfitting of the model) than to define what is right. For data labeling, the users can choose which data regions they intend to annotate for model refinement. In this way, the users can efficiently integrate novel domain knowledge into the system. Labeling of data in regions near the decision boundaries helps increase the classifiers confidence, whereas labeling of data in regions far away from the decision boundary helps explore new regions of the data space and might reduce extensive class confusion. However, users can also rely on the classifier model to provide the most beneficial data instances for labeling utilizing active learning.

In the next sections, we introduce the main components of a visual analytics system for inter-active learning and explain their application for the tasks mentioned above. These components include, besides different (coordinated) views on the data and model, also the definition and implementation of an appropriate classifier model. Furthermore, we will address issues of scalability and input noise that affects the trained model.

4 Classifier Requirements

In this section, we describe the requirements for a learner for ad-hoc training in general and the classifier model we use for video visual analytics in particular.

Filters generated by ad-hoc training have to be of low time complexity because such filters are often used to facilitate scalability with increasing data size. In this paper, we use a *cascade of classifiers* (Fig. 3 (b)) to predict class assignments of sliding windows in each video frame. In combination with a set of basic, yet fast to compute, *rectangle features*, this method has become popular with the work of Viola and Jones [39] in the context of face detection. Rectangle features (Fig. 3 (a)) operate on gray-value images and are computed by subtracting the sum of pixel values of the black part from the sum of pixels of the white part.

Similar to Viola and Jones, we also use a committee of thresholded rectangle features as *weak classifiers* within each node of the cascade. The features for each node are typically selected and weighted by some boosting algorithm, such as *AdaBoost* [9]. The high efficiency of the cascade during evaluation of sliding windows (various rectangular cropped parts of the video frame) results from the low complexity of the first nodes. With only few computations, the first node already rejects about half of all sliding windows; the complexity and number of features typically increase with each node. Sliding windows that pass all nodes in the cascade are considered detections (Fig 3 (b)), such as the sliding window in Fig 3 (a), which contains a person onto whom the rectangle feature selected is superimposed and aligned by boosting. This often applied approach only distinguishes between sliding windows that are "detected" by the cascade or not, hence it defines a binary classification problem. If not stated otherwise, the parameters we use for our cascade of classifier are derived from the original work of Viola and Jones [39].

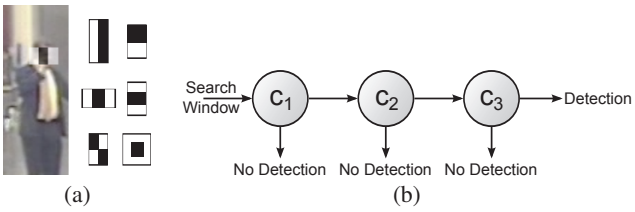

Figure 3: (a) Six prototypical rectangle features and a cropped part of a sliding window containing a detected person onto whom one of the rectangle features superimposed. (b) Each node of the cascade of classifiers makes a binary decision on the sliding window: whether it is dropped or processed by the next classifier node. Sliding windows that process the whole cascade are considered to be detections.

As reported by Tomanek and Olsson [36], it is critical for interactive refinement of classifiers that retraining can be performed very fast to keep the idle time of users at an acceptable level. Fails and Olsen [8] even claim that, in order to be effective, the classifier must be generated from the training examples in under five seconds. These requirements also apply to inter-active learning. Hence, most training algorithms are not suitable for inter-active learning because they tend to need several hours of time for learning the model based on a huge set of labeled data. However, by retraining the classifier with only the samples the users labeled in the current cycle, we can meet the efficiency requirements. Methods that can iteratively update the model according to newly provided data/label pairs are called online training methods.

In this paper, we use a modified version of online AdaBoost, introduced by Oza [23]. Since in online training, the learner is faced with only one example a time, the basic idea of online boosting is to maintain statistics for each feature that capture its performance history for all the samples seen so far. This, however, implies that greedy selection of features cannot consider the complete set of examples when choosing a particular feature, since future examples are not available. Hence, the performance of online AdaBoost approaches the performance of batch-mode training in the limit. Based on the introduced statistics, an estimated error e_i can be calculated for each weak classifier. A weak classifier h_i^{weak} is built from a rectangle feature i that is thresholded after evaluation to obtain a binary decision (i.e., $h_i^{\text{weak}} \in \{-1,1\}$); hence, we use the terms feature and weak classifier interchangeably.

A subset of all available weak classifiers is selected by the greedy boosting algorithm according to the lowest error to form a *strong classifier* h^{strong}. The binary decision about the class assignment of each sliding window s is made in each node of the cascade by its respective strong classifier h^{strong}. Therefore, the weighted sum of binary responses of all n boosted weak classifiers that belong to the strong classifier and threshold t are used:

$$h^{\text{strong}}(s) = \text{sign}(\text{conf}(s) - t)$$

$$\text{conf(s)} = \sum_{i=1}^{n} a_i h_i^{\text{weak}}(s) \tag{1}$$

$$a_i = \log\left(\frac{1-e_i}{e_i}\right)$$

AdaBoost lives on rating the importance of each example for training (a correctly classified example is of lower importance than an

incorrectly classified one). Online AdaBoost estimates the importance of an example for each feature based on the binary decisions of the preceding weak classifiers. According to the estimated importance, the history maintained by each feature is influenced.

Training of a node means to adjust the thresholds, weights, and history of all features. The approach of Oza only uses a fixed number of features. We improve on this by integrating the idea of Grabner and Bischof [12]. They use selectors to dynamically choose the committee of weak classifiers with the best performance out of a number of features that are constantly trained. In this way, changing number of features also becomes possible.

Each strong classifier of the cascade is trained with a modified version of this online AdaBoost algorithm. The main improvements we make to the method of Grabner and Bischof affect the statistics maintained for each strong classifier. By introducing two histograms (one for the positive and one for the negative examples) of the conf measure (see Equation (1)), we enable a cascade construction that is similar to the original algorithm by Viola and Jones. Using the histogram of confidences of positive training examples, the true positive rate can be adjusted to match the classification goals of a node by decreasing the threshold t to an appropriate level. The false positive rate is then accessible by the confidence histogram of negative examples, by summing up the bins between one and the currently chosen threshold. After the cascade node was trained by an example, either the former or the latter histogram, is updated (depending on the label of the data example) by increasing the particular bin by α. Finally, the histogram is normalized to one. The constant α controls the decay rate that is necessary because the stored confidence values become outdated by training. In our examples, we use an experimentally derived value of $\alpha = 2/\eta$, with η being the number of samples seen so far.

To increase robustness for imbalanced numbers of positive and negative training examples, we modify the boosting algorithm to optimize the distance between the receiver operating characteristic (ROC) of the node's classification history and the point of optimal classification (i.e., TPR $= 1$, FPR $= 0$), involving calculation of true positive rate (TPR) and false positive rate (FPR) of each weak classifier. New nodes are added to the cascade until the maximum error rate $\text{FPR}_{\text{overall}} > \text{FPR}_{\text{overall}} = \prod_k^m (\text{FPR}_k)$ is met. For our experiments, we choose an maximum error rate consistent with Viola and Jones [39]: $\text{FPR}_{\text{overall}} = 10^{-5}$. Finally, we enable the users to modify the cascade, by adding new features (user-defined features) to a node or changing the position or shape of features that were selected by the algorithm.

5 Active Learning

When users lack the knowledge which data instances are most efficient to label, they can use active learning. Pressing a button, they automatically obtain a selection of data instances for which the active learner is most uncertain. The selection is shaped by a pool-based uncertainty sampling approach. For the cascade of classifiers we use, uncertainty u about the true label of a sliding window s corresponds to a sum of confidence (Equation (1)) values of all m nodes involved in making the class decision:

$$u(s) = \frac{1}{\sum_{k=1}^{m} \text{conf}_k(s)} \tag{2}$$

This is equivalent to the definition of confidence by Visentini *et al.* [40] and Graber and Bischof [12]. The most uncertain data instances will be selected to be further analyzed by the users.

6 Visualization and Interaction

In this section, we introduce the different views and interaction technique of our inter-active learning framework. The screenshot of the workspace depicted in Fig. 4 shows the three main areas of the graphical user interface (GUI). Left, we see the trained classifier model (Fig. 4 (b)) and the cascaded scatterplot (Fig. 4 (a)) showing the evaluation results of the cascade for performance assessment. On the right, three different views on the model and data are available for annotation and model modification, which are used in the second step of each training cycle. In-between both GUI areas, the current selection of data instances, cascade nodes, and features is shown (Fig. 4 (c)). The selections connect the performance assessment step with the refinement step and, in this way, provide a natural arrangement of tasks. For interpretability, a distinct color is assigned to each type of selection. This color is used to highlight the selection in each of the coordinated views.

6.1 Cascaded Scatterplot

To present the feedback on the quality of the current classifier, we introduce a novel visualization of the class distribution of data instances (sliding windows of the video) in each stage of the classifier. This visualization, which we call *cascaded scatterplot*, integrates multiple dependent scatterplots that are horizontally aligned to match with the cascade information (Figs. 4 (a) and (b)).

In the cascaded scatterplot, the abscissa is divided into m parts, with m being the number of nodes in the cascade. In contrast to conventional scatterplots, cascaded scatterplots represent each data point up to m times. Coordinates of a data point in the cascaded scatterplot depend on the classification quality of the sliding window s by the respective strong classifier (cascade node). The x-value of each instance of a data point is made up of an integer value that determines its assignment to a cascade node as well as of a fractional part that represents the confidence of each classifier's decision on the data point. Hence, data points of two classifiers cannot overlap in their x-value. For each data point and classifier, we plot the normalized confidences (distance to the decision boundary on the x-axis) $c_k(s) \in [-1;1]$ against the feature robustness $\text{r}_\text{k}(s) \in [0;1]$ (y-axis) for each node N_k.

The robustness $\text{r}_\text{k}(s)$ of a feature decision is influenced by two components: the robustness of the feature weights r^{weight} of each node with n_k features, and the distance between the feature response of the signal and the threshold of the feature (its decision boundary) r^{margin}. Together, the feature robustness $\text{r}_\text{k}(s) = \text{r}_k^{\text{weight}}\,\text{r}_k^{\text{margin}}(s)$ indicates the reciprocal of the influence of signal noise on the decision of the strong classifier N_k. The robustness of the feature weights $\text{r}_k^{\text{weight}}$ utilizes the normalized entropy of the feature weights a_i to penalize the skew of weight distribution to a small amount of huge weights because in this case, the decision of a node may be changed only due to the flipping a couple of its features' decisions:

$$\text{r}_k^{\text{weight}} = -\sum_{i \in N_k} \frac{a_i \log(a_i)}{\log(n_k)}$$

Data instances with a small distance between their feature response and the feature's threshold are likely to flip the weak classifier's decision in the presence of noise. Therefore, $\text{r}_k^{\text{margin}}$ includes the average distance (normalized to the maximum margin of the features) between the features' decision boundaries t_i and the signal's response to each feature $f_i(s)$:

$$\text{r}_k^{\text{margin}}(s) = \frac{1}{n_k} \sum_{i \in N_k} \frac{|t_i - f_i(s)|}{\text{maxmargin}_i}$$

The feature robustness is shown along the y-axis of the scatterplot and helps the users judge the classifiers sensitivity to data noise that influences the general quality of the model. On the x-axis, the uncertainty of the class assignment is shown by the distance of the data points to the decision boundary of each strong classifier in the cascade. The confidence measure $c_k(s)$ is normalized to the range

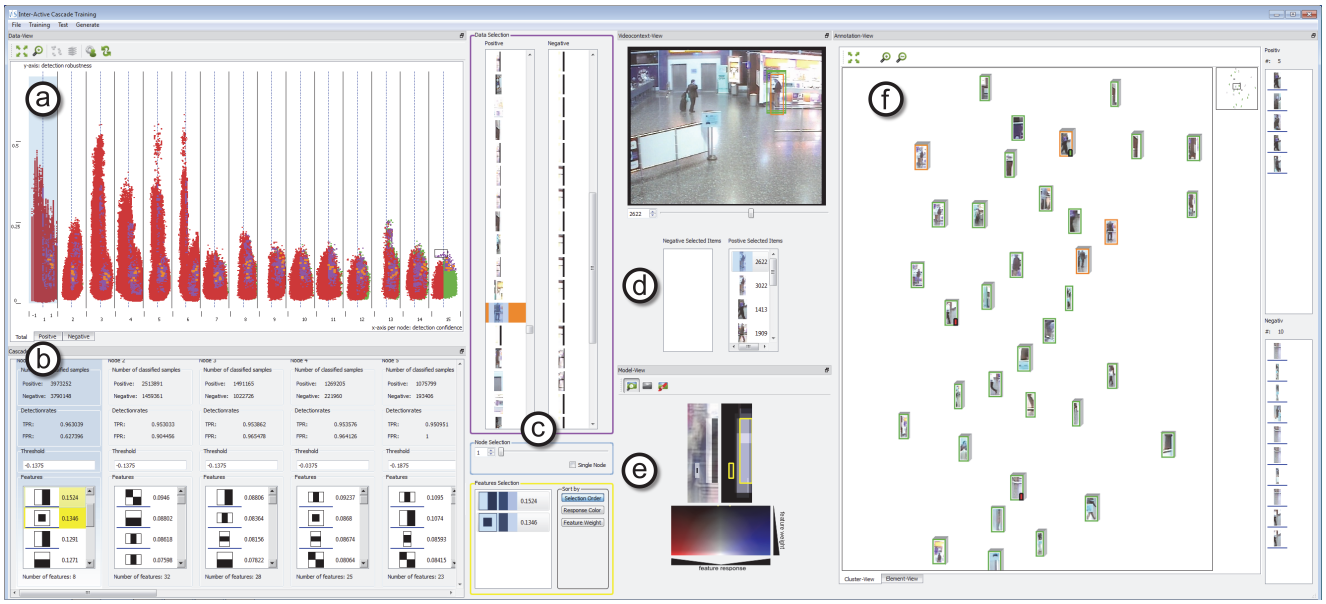

Figure 4: Typical workspace of our visual analytics system for inter-active learning after learning with a couple of training examples: (a) cascaded scatterplot, (b) cascade information, (c) selection interface, (d) video context view, (e) visualization of classifier model, (f) annotation view. Details of the components can be found in Section 6.

of $[-1,1]$, where $c_k(s) = -1$ and $c_k(s) = 1$ represent confident decisions, either negative or positive according the class membership of the data instance:

$$c_k(s) = \frac{\text{conf}_k(s)}{\sum_{i \in N_k} \left| a_i h_i^{\text{weak}}(s) \right|}$$

Besides both boundaries of maximum confidence, we illustrate the decision boundary (dashed lines in Fig. 4 (a)) at $c_k(s) = 0$ for orientation, too. By selecting one or more data points in the cascaded scatterplot representation of a node, the data points will be highlighted in all other nodes in which they appear. In this way, the users can easily assess the class assignment and the quality of the decisions of the selected data throughout the whole cascade. The assignment of each data instance to either the positive class (e.g., person) or the negative class (i.e., background region) is indicated by the respective color (green or red) of the data point in the cascaded scatterplot for each node. For convenience, the users can choose to display only positive or negative classifications, or both combined in one plot.

In combination with other views, the cascaded scatterplot is mainly used to assess the performance of the classifier and to provide feedback on the training progress. In this way, users gain trust in the trained classifier and may decide to stop the refinement process, either because no further progress in training is experienced or the quality of the classifier reached a sufficient level. Recognition of the right point to stop training is of practical relevance, as the survey by Tomanek and Olsson [36] points out: most of the participants had a stopping criteria that would fit the context of ad-hoc training. Furthermore, the cascaded scatterplot is used to select data instances for labeling and thus, to avoid overfitting of the classifier, reduce uncertainty of the classification decision, or improve generalization by exploring new data regions.

To handle the tremendous amount of data we face in the context of video analysis, the amount of data displayed can be adjusted by the user. In this way, we maintain the responsiveness of the interactive visualization and reduce overdrawing in the cascaded scatterplot. The users can define the frame interval and a number of frames to be evaluated. These will be randomly chosen from the interval. Further, we constrain the number of evaluated data instances, by considering only sliding windows that have an aspect ratio similar to the average aspect ratio of the positive training examples.

The displacements dx and dy (measured in pixels (px)) and scaling factor sf of the sliding windows are further restricted. Applying our default values ($dx = 5\,px$, $dy = 5\,px$, and $sf = 1.5$) results in almost two million data instances to be evaluated for 50 frames of a video with $720 \times 576\,px$ resolution. This shows that evaluating an entire video would exceed the capabilities of human and machine very quickly. Hence, we provide further constraints to keep the amount of data at a manageable level. Besides random sampling of a smaller amount of data instances from the video, we use two additional methods to reduce the amount of data displayed. The first method focuses on relevant regions of the data distribution by interactive definition of robustness and uncertainty intervals in the cascaded scatterplot. The second method automatically selects a fixed number of the most uncertain detections by using active learning.

6.2 Annotation

After data instances have been selected, either by the users or automatically by the active learning subsystem, the data points are visualized in the annotation view (Fig. 4 (f)). This helps the user to comprehend the quality of the classifier and discover areas of data instances that have the wrong class assignment (class confusion).

To facilitate overview and efficient annotation of the data instances, we project the data points onto a two-dimensional map according to their similarity. For this purpose, we use the dimension reduction algorithm developed by Van der Maaten and Hinton [38], called t-Distributed Stochastic Neighbor Embedding (t-SNE). Van der Maaten and Hinton showed that this method provides good visualizations of high-dimensional data lying on related low-dimensional manifolds, such as images of multiple objects captured from different viewpoints.

We visualize the two-dimensional map provided by t-SNE by displaying thumbnails of the sliding windows. Class assignment of the classifier as well as data selection is shown by the border color

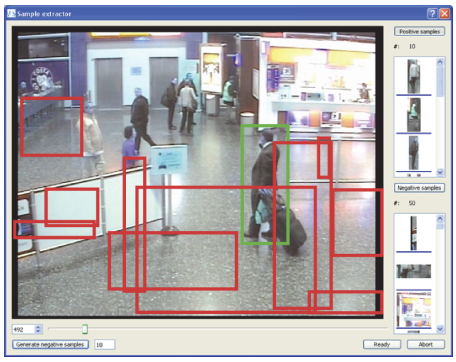

Figure 5: Selection of the initial training set for bootstrapping.

of the thumbnail (red, green, or orange). Additionally, the thumbnails are augmented by the label information if a data instance has already been annotated (small red or green boxes in the right corner of the thumbnails). To reduce visual clutter and to increase annotation speed, overlapping thumbnails are depicted as clusters. Therefore, only the medoid of a cluster is depicted as representative data instance. The clusters are determined by kernel density estimation in screen-space with a fixed bandwidth that is adapted to the size of the thumbnails. The medoids are iteratively selected according to their density. This way, any overlap of thumbnails is avoided. A cue of the cluster size is shown by the size of the shadow dropped by the medoid's thumbnail. Users can navigate through the data by zooming and panning the map. Zooming and panning enables the users to explore particular clusters that will be unfolded step by step and thus a detailed view on their elements is shown. A minimap supports navigation in the data projection.

Similar to Möhrmann *et al.* [20], we allow for fast labeling of multiple data instances by selecting and annotating whole clusters of data instances. This supports the users in one of their main tasks while iteratively refining the classifier. The thumbnails augmented with the classifier decision and the label information therefore enable the users to make quick decisions which data instances should be labeled next. Selecting one or multiple data instances in the annotation view will also select these instances in the other views. Therefore, the annotation view helps select similar data instances for further inspection in the other views.

For annotation, the linked video context view (Fig. 4(d)) is particularly useful because it displays the selected sliding windows in their video frames and, thus, complements the thumbnails of the annotation view by the video context. Furthermore, selection highlighting in the cascaded scatterplot helps the users determine the robustness and confidence of the classifier's decision. By this process, the users can quickly select and label the data that is most beneficial for further training.

6.3 Model Representation and Manipulation

Understanding the classifier model can be crucial to assess its quality and to determine further actions to increase the classifier's performance. Model understanding becomes beneficial, especially in cases where only a small amount of labeled data is available and, thus, cross validation is not applicable.

A famous example of classifier failure due to a restrictive dataset is the often-cited story of an artificial neural net trained by the army to detect tanks [7]. After first success on training and validation sets, the system totally failed on a newly captured validation set. After investigating the issue, it turned out that the system learned to distinguish between cloudy and sunny skies rather than between tanks and bushes, due to a bias in the data sets. Recent discussion exhibited that bias is common to most datasets used to train computer vision systems [37], even though the datasets strive for representing the whole visual world. Unfortunately, the bias of datasets in ad-hoc training is presumably disproportionately stronger than in such carefully assembled datasets. Due to such bias, it is important to understand the trained model for verification and validation, e.g., by using visualization [33]. We therefore visualize the classifier model and let the users investigate the trained model and its behavior, as often suggested [16].

One or multiple data instances, selected in the annotation view or in the cascaded scatterplot, are depicted in the model visualization (Fig. 4 (e)). If multiple data instances are selected, the mean image of their sliding windows is displayed. This outlines their commonalities that become recognizable as patterns of patterns in the mean image [42]. This perspective, however, is similar to the view the classifier has on the data. Next to the mean image, the corresponding feature response map is depicted. This map is constructed by superimposing the color-encoded feature response and feature weight. A legend of the bimodal red-blue color mapping of feature response and the luminance mapping of the feature weight is depicted in Fig. 4 (e). In combination, these two maps help understand the relevance of different parts of the model to the classifiers and, possibly, the reason for that importance.

To further facilitate model understanding, users may select a single node or multiple nodes of the cascade to be evaluated and depicted in the model view. Further control of the visualization is provided by selecting arbitrary features by drag & drop from the cascade information view (Fig. 4 (b), yellow selections). The selected rectangle features will be displayed in a list with their according weight and response to the selected data instances. These values are also marked in the legend. Additionally to that, outlines of the features—with respect to their position and scale in their classifier node—augment the aggregated feature response map and the mean image of data instances. This supports the users in determining the influence of each feature and the structures it detects.

Based on the model visualization, the users may experience discrepancy between their idea of the intended filter and the trained classifier model. In this case, the users are provided by several tools to alleviate the discovered discrepancy by integrating more background knowledge into the classifier. The users may change the properties of one or more features, such as their type, scale, location, and threshold. Furthermore, the users may add or remove feature to or from arbitrary nodes. Finally, by brushing on the mean image, the users can define areas in which the classifier is not allowed to place any features in. After modification of the classifier, the new model will be automatically refined in the next cycle.

Modification of the model based on the users' expertise can be used to boost the learner, especially in early training cycles, by including fresh domain knowledge. Overfitting of the learner as well as model errors due to lack of generalizing training data (i.e., the samples are not independent and identically distributed) can be detected and corrected (e.g., by removing specialized features, which is similar to pruning a decision tree). By locking particular regions of the sliding windows from being evaluated by features, insignificant areas, such as background or potentially occluded regions can be excluded from being regarded in training. In this way, training has to focus on more relevant parts of the training data (e.g., tanks instead of skies).

Please note that we do not suggest defining the classifier model from scratch because this is often not possible in complex data domains. Nevertheless, direct model manipulation can be beneficial to integrate background knowledge that is not available in the form of labeled data instances, in cases where discrepancy between trained classifier model and the intended filter was experienced.

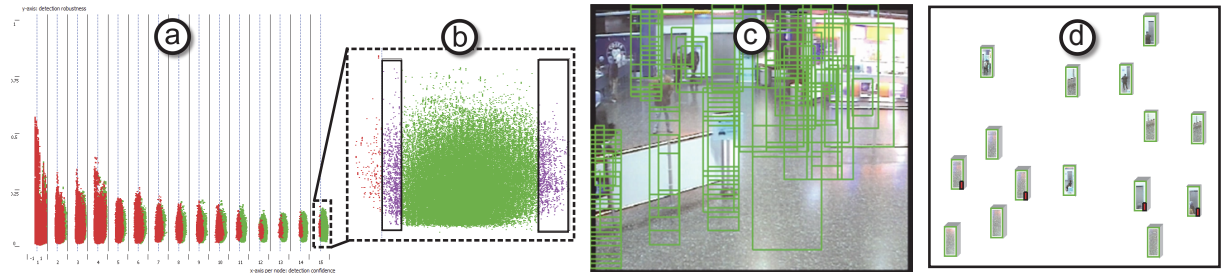

Figure 6: (a) Visual feedback of the classifier's performance after bootstrapping with 25 positive and 50 negative examples. The high number of false positive detections is indicated by the large amount of positively classified data points (green dots) in the last cascade node. (b) Selection of data points that are close and far from the decision boundary. (c) The distribution of sliding windows classified as people (green rectangles). (d) Labeling of false positive data instances in the annotation view.

7 Usage Scenario

In this section, we provide an exemplary usage scenario of our interactive learning system to demonstrate its capabilities and potentials[1]. We keep the example simple by learning a classifier model in only four training cycles; this classifier is capable of detecting persons in video sequences. This classifier represents an example of a filter that may be included in an analytical reasoning process and, thus, being defined under severe time constraints. For simplicity reasons, we only depict the most relevant views and elements of the GUI. We use two video sequences gathered from the i-Lids multi camera tracking dataset[2], one for training and one for validation of the classifier. To verify our method, we compare its performance with training the initial classifier (bootstrapped in Cycle 1) using random sampling of annotated ground truth data instances. Further, we provide the performance of active uncertainty sampling, according to the measure introduced in Equation (2). For both uncertainty sampling and random sampling, we choose a balanced set of positive and negative examples.

Cycle 1: A first training set is populated, used to bootstrap the ad-hoc classifier. We select just 25 positive examples by drawing rectangles on the video frames, as illustrated in Fig. 5. Additionally, 50 negative examples are also selected, either by manually defining the regions or by drawing random samples from the video, in which only small overlap with positive examples is allowed. Both positive and negative examples are depicted in the video player and in separate lists for visual inspection by the user. Based on these examples, the initial classifier cascade is automatically trained.

Cycle 2: A first glance at the cascaded scatterplot (Fig. 6(a)) exhibits a high false positive rate by the large number of positively (green dots) classified data instances in the last node. By browsing the video in the context view, this assumption is confirmed: the green rectangles in Fig. 6(c) represent sliding windows classified as persons. We choose to label some misclassified background patches using the annotation view. First, we select the set of positive classified data instances in the last cascade node, since we are only interested in the sliding windows that are (false-) positively classified by the whole cascade of classifiers. However, the labeling of these instances affects all nodes, such that also earlier nodes adapt to important new patterns. We choose to select some data instances near the decision boundary and some samples that are far away from the decision boundary for annotation (Fig. 6 (b)). The first selection includes the data instances the model is most uncertain about their classification. This is similar to uncertainty sampling in active learning. Selection of data instances far away the decision boundary includes the data instances the model is most certain about, but this does not mean that these instances are assigned to the right class. Due to the very low number of initially labeled examples it is very likely that we find patches containing no person in the region far away from the decision boundary. By labeling data instances of those both regions, we expect to improve the classifier model most. Within the active learning domain, such query of data instances far from the decision boundary is sometimes termed exploration [22]. For labeling, we navigate the annotation view (by panning and zooming) to a region of clusters that contains various instances of misclassified samples (see Fig. 6 (d)). Further zooming reveals the individual members of the clusters. By selecting individual instances or whole clusters, the video context view jumps at the position of the video from which the selected data instance or the medoid of the cluster were sampled and represents its sliding window by a rectangle in the selection color (orange). This way, the spatio-temporal context of the data instances becomes accessible to the users. Next, we add 50 instances to the set of negative training examples by selecting clusters or single instances. The thumbnail images in the annotation view are augmented by red rectangles to indicate which data instances have been labeled. Finally, we retrain the classifier and proceed with the next refinement cycle.

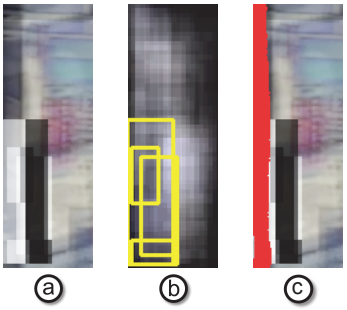

Figure 7: Model inspection and manipulation during the 3rd cycle. (a) The mean image of false positive examples exhibits a black edge pattern of patterns at the left border that is superimposed by rectangle features that response to this pattern. (b) The response map confirms the impact of the left boundary region to person classification. (c) Model manipulation prevents features to be aligned with the edge pattern, since it is deemed to be not relevant for people detection (integration of background knowledge).

Cycle 3: The cascaded scatterplot and the model view of the cascade provide visual feedback about the last training cycle. We spot decrease of the number of positive classifications. However, we experience by browsing the classification results that the classifier is not satisfactorily trained, since still many background regions are

[1]The supplementary material includes a video of the usage scenario. http://www.vis.uni-stuttgart.de/index.php?id=vva

[2]http://www.homeoffice.gov.uk/science-research/hosdb/i-lids/

considered as persons. Hence, we decide to label some false positive examples again.

During the annotation of 50 false positive data instances (and one false negative), we inspect the model representation for common reasons of the wrong classification of background data instances. The mean image of several false positive samples quickly exhibits a *pattern of patterns*, an explicit vertical edge at its left border (cf. Fig. 7 (a)). Superimposing the mean image with rectangle features selected by the boosting algorithm reveals the importance of the vertical edge pattern for recognition. This importance is further confirmed by the strong alignment of some of the rectangle features with the edge and the feature response map in Fig. 7 (b). After we identified the adaption of the classifier to this pattern, which presumably stems from different wall colors and the black border surrounding the video, we decide to alleviate its effect. Therefore, we manipulate the model by defining an area in which the placement of rectangle features by the training algorithm is prohibited. This area is illustrated by the red region in Fig. 7 (c).

Cycle 4: After retraining the classifier, we investigate the training state of the classifier (cf. Fig. 8). It turns out that there is a decay of the number of negative examples rejected by the final cascade nodes, whereas the false positive rate determined by the seen training samples increases. Since this can be seen as stopping criterion, we terminate the training process at this point.

Figure 8: Since the number of rejected negatives decrease, while the false positive rate increases in the final cascade nodes, we deem the training process to be finished.

Discussion: Comparison of the performance of inter-active learning with random sampling and uncertainty sampling in Fig. 9 reveals the benefit of direct knowledge integration by inter-active learning in the context of ad-hoc classifier training. By only labeling about 100 additional data instances, we were able to achieve within 4 cycles a classification performance that is comparable to the results of the other learning methods that settle down with the number of data instances exceeding 500-600 samples. After a first cycle of initial training, Fig. 9 shows that the annotation of 50 false positive samples in Cycle 2 successfully decreases the false positive rate. However, also the number of true positive detections is reduced, since just negative samples were considered. Model manipulation and additional data annotation in the third cycle finally provide us with a performance comparable to the best achievable results for this complex problem. This means for better results, a more powerful model has to been chosen. Further, Fig. 9 reveals a possible problem of uncertainty sampling in ad-hoc training situations. Its true positive rate suddenly drops when querying labels for the most uncertain (i.e., near the decision boundary) data instances. Since this behavior is only observable in the context of uncertainty sampling, we presume this effect to originate from the query function that introduces a bias to the distribution of data instances.

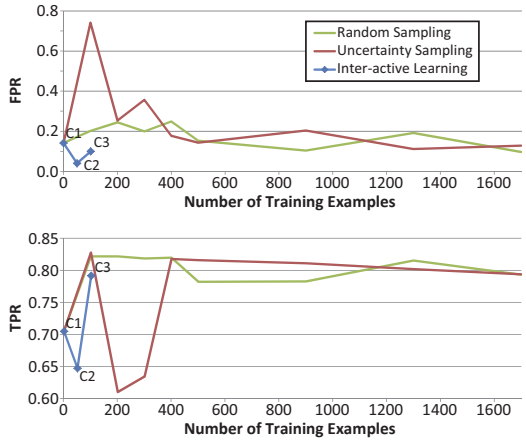

Figure 9: Comparison of the performance of inter-active learning, random sampling, and uncertainty sampling with respect to the number of annotated data instances. Inter-active learning requires only about 100 additional labels to achieve a similar performance limit as the other methods achieve, when using 500-600 labels.

8 Conclusion and Future Work

In this paper, we have introduced inter-active learning, a method that extends active learning to a visual analytics process in order to define filters by ad-hoc training classifiers. We have outlined the major challenges of ad-hoc training and presented a way to master them by tightly coupling human expertise with machine learning. The main aspects of our approach are: the quality assessment and model understanding by explorative visualization, the integration of experts' background knowledge by data annotation and model manipulation, and the use of automatic methods to support the users in refining the classifier model.

We have been able to demonstrate the power of our method by a usage scenario in which we have compared inter-active learning with active learning and passive learning (random sampling of training data). The usage scenario has exhibited the advantages and possibilities inter-active learning provides.

Although we have been able to show the benefits of inter-active learning, more research is required to investigate the extent of practical applications and to judge its efficiency in realistic scenarios. A question that arises in this context is the required proficiency of the human experts. Is domain knowledge sufficient to utilize inter-active learning, or to which extent are skills in machine learning necessary? Also, further development of new, and improvement of existing, components for inter-active learning is required, especially for scalable visualizations, classifier models, and integration of automatic methods and techniques for semi-supervised learning. We have demonstrated inter-active learning in the context of video visual analytics; it remains an open question how successfully it can be applied to other domains. This would also require a more comprehensive evaluation of inter-active learning.

As we see, various exciting directions are open for future research, including the inspection of the bias introduced by inter-active learning into the classifier model, such as annotation bias by using cluster-based labeling tools, as well as the integration of fuzzy model modification and data annotation that account for the uncertainty of the human expert.

Acknowledgements

This work was funded by German Research Foundation (DFG) by the Priority Program "Scalable Visual Analytics" (SPP 1335).

REFERENCES

[1] M. Ankerst, M. Ester, and H. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 179–188, 2000.

[2] J. Baldridge and A. Palmer. How well does active learning actually work? Time-based evaluation of cost-reduction strategies for language documentation. In *Conference on Empirical Methods in Natural Language Processing*, volume 1, pages 296–305. Association for Computational Linguistics, 2009.

[3] B. Becker, R. Kohavi, and D. Sommerfield. Visualizing the simple bayesian classifier. In U. Fayyad, G. G. Grinstein, and A. Wierse, editors, *Information visualization in data mining and knowledge discovery*, chapter 18, pages 237–249. Morgan Kaufmann Publishers Inc., 2001.

[4] E. Bertini and D. Lalanne. Surveying the complementary role of automatic data analysis and visualization in knowledge discovery. In *ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery*, pages 12–20, 2009.

[5] D. Caragea, D. Cook, and V. Honavar. Gaining insights into support vector machine pattern classifiers using projection-based tour methods. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 251–256, 2001.

[6] C. Chen. Top 10 unsolved information visualization problems. *Computer Graphics and Applications*, 25(4):12–16, 2005.

[7] H. Dreyfus and S. Dreyfus. What artificial experts can and cannot do. *AI & Society*, 6(1):18–26, 1992.

[8] J. Fails and D. Olsen Jr. Interactive machine learning. In *International Conference on Intelligent User Interfaces*, pages 39–45, 2003.

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[10] Y. Freund, H. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

[11] C. Gasperin. Active learning for anaphora resolution. In *NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 1–8, 2009.

[12] H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 260–267, 2006.

[13] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. *Advances in Neural Information Processing Systems (NIPS)*, 20:593–600, 2008.

[14] F. Heimerl, S. Koch, H. Bosch, and T. Ertl. Visual classifier training for text document retrieval. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visual Analytics Science and Technology 2012)*, 18(12), 2012.

[15] M. Höferlin, B. Höferlin, D. Weiskopf, and G. Heidemann. Uncertainty-aware video visual analytics of tracked moving objects. *Journal of Spatial Information Science*, 2:87–117, 2011.

[16] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann. *Mastering The Information Age-Solving Problems with Visual Analytics*. Eurographics Association, 2010.

[17] D. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *International Conference on Information Visualization (IV)*, pages 9–16, 2006.

[18] T. May and J. Kohlhammer. Towards closing the analysis gap: Visual generation of decision supporting schemes from raw data. *Computer Graphics Forum*, 27(3):911–918, 2008.

[19] M. Migut and M. Worring. Visual exploration of classification models for risk assessment. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 11–18, 2010.

[20] J. Möhrmann, S. Bernstein, T. Schlegel, G. Werner, and G. Heidemann. Improving the usability of interfaces for the interactive semi-automatic labeling of large image data sets. In *Human Computer Interaction. Design and Development Approaches*, pages 618–627, 2011.

[21] F. Olsson. A literature survey of active machine learning in the context of natural language processing. Technical report, Swedish Institute of Computer Science, 2009.

[22] T. Osugi, D. Kim, and S. Scott. Balancing exploration and exploitation: A new algorithm for active machine learning. In *IEEE International Conference on Data Mining*, pages 330–337, 2005.

[23] N. C. Oza. Online bagging and boosting. In *International Conference on Systems, Man and Cybernetics*, volume 3, pages 2340–2345, 2005.

[24] M. Saar-Tsechansky and F. Provost. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2):153–178, 2004.

[25] C. Seifert and M. Granitzer. User-based active learning. In *IEEE International Conference on Data Mining Workshops*, pages 418–425, 2010.

[26] C. Seifert, V. Sabol, and M. Granitzer. Classifier hypothesis generation using visual analysis methods. In *Networked Digital Technologies*, volume 87 of *Communications in Computer and Information Science*, pages 98–111. Springer, 2010.

[27] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin, Madison, 2009.

[28] B. Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Conference on Empirical Methods in Natural Language Processing*, pages 1467–1478, 2011.

[29] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, 1994.

[30] B. Shneiderman. Inventing discovery tools: Combining information visualization with data mining. *Information Visualization*, 1(1):5–12, 2002.

[31] V. Sindhwani, P. Melville, and R. Lawrence. Uncertainty sampling and transductive experimental design for active dual supervision. In *International Conference on Machine Learning*, pages 953–960, 2009.

[32] J. Talbot, B. Lee, A. Kapoor, and D. Tan. EnsembleMatrix: Interactive visualization to support machine learning with multiple classifiers. In *Conference on Human Factors in Computing Systems*, pages 1283–1292, 2009.

[33] B. Taylor, M. Darrah, and C. Moats. Verification and validation of neural networks: A sampling of research in progress. In *International Society for Optics and Photonics (SPIE)*, volume 5103, pages 8–16, 2003.

[34] S. Teoh and K. Ma. PaintingClass: Interactive construction, visualization and exploration of decision trees. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 667–672, 2003.

[35] J. Thomas and K. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2005.

[36] K. Tomanek and F. Olsson. A web survey on the use of active learning to support annotation of text data. In *NAACL HLT Workshop on Active Learning for Natural Language Processing*, pages 45–48, 2009.

[37] A. Torralba and A. Efros. Unbiased look at dataset bias. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1521–1528, 2011.

[38] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[39] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[40] I. Visentini, L. Snidaro, and G. L. Foresti. On-line boosted cascade for object detection. In *International Conference on Pattern Recognition (ICPR)*, pages 1–4, 2008.

[41] M. Wang and X. Hua. Active learning in multimedia annotation and retrieval: A survey. *ACM Transactions on Intelligent Systems and Technology*, 2(2):10:1–10:21, 2011.

[42] C. Ware. *Visual Thinking for Design*. Morgan Kaufmann, 2008.

[43] M. Ware, E. Frank, G. Holmes, M. Hall, and I. Witten. Interactive machine learning: Letting users build classifiers. *International Journal of Human-Computer Studies*, 55(3):281–292, 2001.