The Visual Computer Journal manuscript No.
(will be inserted by the editor)

Online Learning and Detection of Faces with Low Human Supervision

M. Villamizar, A. Sanfeliu and F. Moreno-Noguer

Received: date / Accepted: date

Abstract We present an efficient, online, and interactive approach for computing a classifier, called Wild Lady Ferns (WiLFs), for face learning and detection using small human supervision. More precisely, on the one hand, WiLFs combine online boosting and extremely randomized trees (Random Ferns) to compute progressively an efficient and discriminative classifier. On the other hand, WiLFs use an interactive human-machine approach that combines two complementary learning strategies to reduce considerably the degree of human supervision during learning. While the first strategy corresponds to query-by-boosting active learning, that requests human assistance over difficult samples in function of the classifier confidence, the second strategy refers to a memory-based learning which uses κ Exemplar-based Nearest Neighbors (κENN) to assist automatically the classifier. A pre-trained Convolutional Neural Network (CNN) is used to perform κENN with high-level feature descriptors. The proposed approach is therefore fast (WilFs run in 1 FPS using a code not fully optimized), accurate (we obtain detection rates over 82% in complex datasets), and labor-saving (human assistance percentages of less than 20%).

As a byproduct, we demonstrate that WiLFs also perform semi-automatic annotation during learning, as while the classifier is being computed, WiLFs are discovering faces instances in input images which are used subsequently for training online the classifier. The advantages of our approach are demonstrated in synthetic and publicly available databases, showing comparable detection rates as offline approaches that require larger amounts of handmade training data.

M. Villamizar
Idiap Research Institute, Switzerland.
E-mail: michael.villamizar@idiap.ch

A. Sanfeliu
Institut de robotica i informatica industrial, Barcelona, Spain.
E-mail: sanfeliu@iri.upc.edu

F. Moreno-Noguer
Institut de robotica i informatica industrial, Barcelona, Spain.
E-mail: fmoreno@iri.upc.edu

1 Introduction

In recent years we have witnessed the impressive growth of the amount of visual data available on Internet and specialized databases such as images and videos. However, most of this data is either raw data lacking of any kind of annotation or data partially labeled.

Computer vision, and particularly object recognition, is one of the research fields that has benefited greatly from this abundance of visual data. Nowadays, there exist relatively large datasets containing diverse objects in images and videos which are used to compute and evaluate object recognition methods, and as benchmarks among different approaches [9,20,43]. Nevertheless, the computation of these databases becomes a tedious and time-consuming task since they require trained humans to annotate a large number of object instances in images. Recently, crowdsourcing initiatives have emerged to mitigate these demanding tasks[1].

A wide variety of statistical methods have been proposed for object recognition using these databases in the last years. Machine learning techniques like Support Vector Machines [8,10,35], Boosting [3,31,49,50,56], Random Forests [7, 13,39], and more recently Deep Learning [19,27,40] are commonly used to build classifiers robust to large intra-class variations and other artifacts such as loss of image resolution, rotations, deformations or lighting changes.

Despite the outstanding results achieved by these classifiers, most of these methods are not suitable to compute object classifiers from dynamic data streams currently available (e.g, images acquired from Internet or image sharing websites[2]) because they are usually built offline after processing all training data and taking whatever amount of time. As a result, these methods can not be applied to situations in which either the training data is obtained continuously, or the object appearance changes over time.

In such cases, we need to resort to online learning techniques to simultaneously compute and update the classifier as new data becomes available. Indeed, this has been already explored in several works [4,14,15,18,23,25,44,58]. Yet, most of these approaches are focused on detecting or tracking single object instances and exploiting temporal coherence. The classifiers built this way become too specialized, and generally are constrained to situations where the object appearance just changes smoothly between consecutive frames. Otherwise, these classifiers tend to suffer from

[1] https://www.mturk.com/mturk/welcome
[2] https://www.flickr.com/

Fig. 1 Wild Lady Ferns (WiLFs) for online learning and detection of faces in images. The proposed method discovers new face instances (green boxes) while computing a discriminative classifier with small human supervision. Red boxes make reference to false positives whereas black ones are the ground truth. The online classifier is trained with samples annotated by the human (indicated by the letter H at the top left of images) or by the machine (letter M) using κ Exemplar-based Nearest Neighbors (κENN). Exemplar samples are denoted in images by the letter X.

drifting[3] because they are built via self-learning. To address this problem, some methods have resorted to human intervention to disambiguate the class labels of difficult training samples [11,52,54,57]. However, these methods usually present moderate levels of human assistance.

In contrast to previous methods, we introduce an approach to incrementally compute a category-level classifier from continuous but not temporally coherent images. In this paper, we focus on the problem of simultaneously performing online face learning and detection, see Fig.1. We refer to our approach as Wild Lady Ferns (WiLFs), in analogy to a variety of ferns that is very resistant to different environments and requires little human assistance[4]. Particularly, WiLFs are an online version of the Boosted Random Ferns (BRFs) classifier [50,51], based on Random Ferns [39] computed over Histograms of Oriented Gradients (HOG) [8].

In order to achieve online performance, we extend the original BRFs with three main components: 1) a bootstrapping strategy to retrain the classifier at run time; 2) an efficient version of the online boosting proposed in [15,51] that allows for a near real time implementation with unoptimized code; and 3) an interactive human-machine method that integrates κ Exemplar-based Nearest Neighbors (κENN) [33] and query-by-boosting active learning [2] to reduce significantly the degree of human supervision, while maintains remarkable recognition rates, since the computation of the classifier is done with training samples annotated both by the human and the machine (κENN).

While previous online techniques were mainly focused to tracking specific object instances, WiLFs are suitable for a much wider range of applications. In the results section we will use standard benchmarks to demonstrate that our classifier is as discriminative as its offline counterparts. We will also show that WiLFs are effective for automatic labeling of large datasets in an interactive manner, where the user just needs to annotate a small fraction of all samples (i.e, those samples with a high class uncertainty) and the classifier discovers automatically most object instances present in the dataset images. We believe that building such robust classifiers with so little effort opens a wide range of applications, for modeling unknown or unexplored areas in a matter of seconds.

[3] The term *drifting* refers to the deterioration of the classifier over time because it is updated with noisy and misclassified samples.

[4] http://www.hardyferns.org/

2 Related Work and Contributions

There is a vast literature on the computation of classifiers for object and face detection. We next highlight only those works related in type and scope to the proposed method, and divide them according to what we think are the three most outstanding contributions of this work.

Efficient categorization. Nowadays, there are many works focused on learning and detecting complete object categories instead of recognizing object instances. Yet, as the time to train is not critical for most of these works, they tend to use offline learning approaches, multiple cues and complex features which are costly to compute, and very large training sets to come up with highly discriminative and robust category-level classifiers [8,10,13,19,27,29,31,35,38,56]. Consequently, these methods are not suitable for online and real-time tasks because training the classifiers can take hours or even weeks in modern PCs or GPUs. In addition, these offline classifiers can not be adapted to new object appearances unless they are retrained from scratch.

Some methods that are able to achieve good recognition rates with moderate intra-class variability and at low computational cost are based on the use of Random Forest and

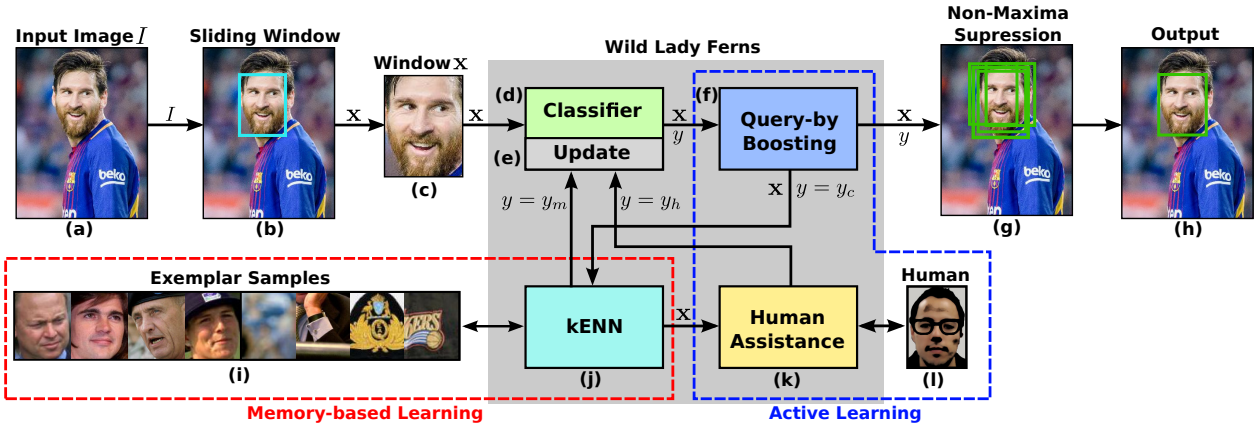

Fig. 2 General scheme of the proposed online detection method based on WiLFs. Given an input image (a), the method performs sliding window over this image (b). Each window sample $\mathbf{x}$ is fed to the online classifier that returns a class estimate y (c,d). If the classifier is uncertain about this estimate (the sample falls inside an uncertainty classification region δ), the method performs active learning in order to label this sample using human interplay (f,k,l), and to update the classifier with this sample (e). Yet, $\mathcal{K}$ exemplar-based nearest neighbors ($\mathcal{K}$ENN) are used in advance to reduce the degree of human assistance over time (j). $\mathcal{K}$ENN predict the sample class from the similarity between the input sample and the exemplar sample set (i). Finally, non-maximum suppression is used to remove multiple hypotheses (g), resulting in the final detection output (h).

Ferns [7,13,23,28,39,41,48]. In particular, the closest work with the presented method is the BRFs classifier which uses boosting to compute an efficient and reliable object classifier from the selection of the most discriminative ferns over HOG [34,50,51].

Contrary to previous works, we propose an efficient and online classifier for object class detection, named as Wild Lady Ferns (WiLFs), that simultaneously learns and detects faces (our target in this paper) in images in 1 FPS using CPU only. This allows to exploit large and dynamic data streams without the need to retrain the classifier from scratch.

Online learning. In order to perform object recognition on the fly, approaches based on online classifiers have been extensively proposed in the past [4,14,15,23,55]. However, such approaches have been designed mainly for the development of robust and adaptive trackers for particular objects that exploit temporal consistency and that only accept small changes in their appearance. As a consequence, these methods fail for category-level detection with large intra-class variability. Typically, these online classifiers are computed via self-learning using their own detection predictions to update and refine the classifiers [14,15,55]. Although this sort of learning allows to compute object models incrementally without using human annotations about the object location, self-learning is prone to *drifting*[3] and thus it deteriorates the performance of the classifier. To cope with this problem, methods commonly use object model priors [17], temporal consistency [15,23], and human assistance to disambiguate those difficult cases where the classifier is uncertain [54,57].

In this paper, WiLFs use an online version of the BRFs classifier [51] but adding human interplay to reduce the risk of *drifting*. This differs from [51] where this online classifier was computed via supervised learning. Additionally, this work provides an extended description and evaluation of the online BRFs classifier for interactive learning tasks. Fig. 2 shows an overview of the proposed method. More specifically, we use an efficient version of online boosting [15] to select iteratively from a shared pool of ferns the most relevant ones to build the classifier (Fig. 2-d). This contrasts with other works based on ferns where they remain fixed throughout the training [39,52,54]. As a result, WiLFs are improved progressively with new available data given that the classifier is adapted at each time instance to new and unknown faces images and imaging conditions.

Slight human supervision. Active learning techniques are widely used in computer vision to reduce the number of training samples that need to be annotated when computing a classifier [46]. Approaches such as uncertainty-based sampling [30] and query-by-committee [47] close the learning loop using human assistance. In these works, the human user acts as an oracle that annotates/labels those samples that the classifier is undecided about their class prediction.

Most existing active methods use pool-based sampling where the learning algorithm iterates repeatedly over a fixed database [30,47]. However, this is impractical for real-world tasks such as crowdsourcing where the unlabeled samples arrive sequentially in the form of continuous rapid streams. Moreover, performing exhaustive search in a data pool is time-consuming, and thus unsuitable for supporting on-the-fly interactive learning [6]. As a result, we opt for a stream-based interactive learning approach where human and machine assistances are used jointly to compute robust object classifiers with scant human supervision over continuous but not temporal coherent streams of images[5]. The proposed ap-

[5] In this learning methodology, the training samples are individually fed to the classifier without seeing them again.

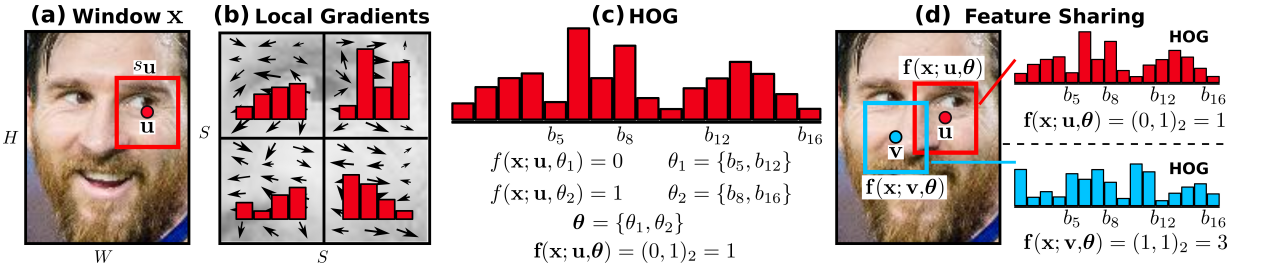

Fig. 3 HOG-based ferns. (a) Image window $\mathbf{x}$ with a subwindow $s_{\mathbf{u}}$ (centered at $\mathbf{u}$) where the fern $\mathbf{f}(\mathbf{x};\mathbf{u},\boldsymbol{\theta})$ is computed. (b) Computation of the local HOG from oriented gradients within $s_{\mathbf{u}}$. (c) HOG-based fern computation with $M = 2$ binary features. (d) Shared features parameters.

proach integrates active learning with memory-based learning (see Fig. 2).

For active learning, we utilize query-by-boosting [2] in order to compute the classifier with difficult samples located nearby the decision boundaries, and to use human assistance to label these samples (see Fig. 2-f,k,l). This results in a more discriminative classifier and in a significant reduction of human annotations. Moreover, we add an adaptive uncertainty threshold δ for active learning so as to reduce gradually the number of human interventions in function of the classifier performance. This differs from other works where the uncertainty threshold is kept fixed during training [52].

For large data sets, though, levels of human intervention can be relatively high. Consequently, we resort to a memory-based method to assist the classifier before humans do, observe Fig. 2-j. We use κ Exemplar-based Nearest Neighbors (κENN) [33] to label automatically new difficult samples using a small subset of exemplar samples learned during learning (Fig. 2-i). That way the amount of human supervision is further decreased but maintaining high recognition rates. Although similar ideas have been applied with success for face detection in the last years [31,29], they were used for offline methods keeping fixed the set of exemplar samples in run time.

In order to perform κENN with difficult image samples nearby the decision boundary, we make use of a deep neural network[6] trained on the Imagenet database [9] to encode these samples with high-level feature descriptors (e.g feature embedding). Despite WiLFs use a deep Convolutional Neural Network (CNN) to disambiguate hard samples, WiLFs are still efficient since the network was computed before (i.e pre-trained network) and because it runs in CPU with low computational cost. Moreover, the network is only used for a few image windows (difficult samples), whereas the random ferns classifier is tested densely over the entire image via sliding window to perform face detection.

3 Wild Lady Ferns: WiLFs

We next describe the main ingredients for the computation of WiLFs: 1) HOG-based ferns; 2) an online boosting algorithm for optimal feature selection, made efficient by means of a feature sharing scheme; and 3) a human-machine interactive learning approach to compute the online classifier with limited annotation cost.

3.1 HOG-based Ferns

Like in BRFs [50], we build the ferns computing binary features in the HOG domain. This has the benefits of bringing both invariance to lighting changes and intra-class variations, while they can be computed very efficiently. More precisely, each fern $\mathbf{f}$ consists of a set of M local binary features f,

$$\mathbf{f}(\mathbf{x};\mathbf{u}_i,\boldsymbol{\theta}_i) = [f(\mathbf{x};\mathbf{u}_i,\theta_1),\ldots,f(\mathbf{x};\mathbf{u}_i,\theta_M)] \quad (1)$$

where $\mathbf{x} \in \mathbb{R}^{H,W}$ is an image window (see Fig. 3-a), and $\mathbf{u}_i$ and $\boldsymbol{\theta}_i$ are the parameters of the fern. The parameter $\mathbf{u}_i \in \mathbb{R}^2$ refers to the 2D location in $\mathbf{x}$ where the fern is evaluated, and $\boldsymbol{\theta}_i = \{\theta_1,\ldots,\theta_M\}$ corresponds to the set of features parameters.

The output of the fern is an M-dimensional binary array, which is in practice represented by an integer value $z \in [0,\ldots,2^M - 1]$. Similarly, the output of each feature $f(\mathbf{x};\mathbf{u}_i,\theta_j)$ on the sample $\mathbf{x}$ is a binary value that captures the difference between the values of two HOG bins chosen randomly. This can be expressed as:

$$f(\mathbf{x};\mathbf{u}_i,\theta_j) = \mathbb{I}(\text{HOG}(s_{\mathbf{u}_i},b) > \text{HOG}(s_{\mathbf{u}_i},b')) \quad (2)$$

where $\theta_j = \{b,b'\}$ are the HOG bin indices, $\mathbb{I}(e)$ is the indicator function[7], and $s_{\mathbf{u}}$ is a $S \times S$ subwindow inside the image window $\mathbf{x}$ where the local HOG is computed (centered at the pixel position $\mathbf{u}$), see Fig. 3-a. $\text{HOG}(s_{\mathbf{u}},b)$ is the value at the b-th bin of the HOG computed in $s_{\mathbf{u}}$.

By way of illustration, Fig 3-b,c shows an example of how a HOG-based fern is computed on a subwindow $s_{\mathbf{u}}$. In this case, $M = 2$ binary comparisons of HOG cells are considered, with individual outputs 0,1. The overall output of this fern is $z = (01)_2 = 1$. The HOG computation is carried out by calculating gradients within $s_{\mathbf{u}}$ and casting votes for a particular spatial partition and orientation bin. Votes are

[6] http://www.vlfeat.org/matconvnet/pretrained/

[7] The indicator function $\mathbb{I}(e) = 1$ if e is true, and 0 otherwise.

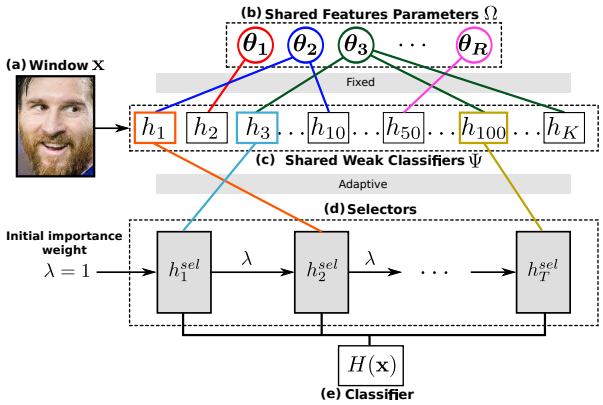

Fig. 4 Computation of the online classifier. (a) Input window $\mathbf{x}$. (b) Shared set of fern-features parameters Ω. (c) Shared set of weak classifiers Ψ using recursively Ω. (d) Concept of selectors used to pick up the best weak classifiers. (e) Assembling of final classifier $H(\mathbf{x})$.

weighted according to the gradient magnitude. For this example, a 2×2 spatial grid and 4 orientation bins are considered. The resulting HOG-descriptor is then a concatenation of local and adjacent distributions of oriented gradients.

3.2 The Online Classifier

3.2.1 Shared Features Parameters

Before describing in detail the computation of the online classifier, we introduce a feature sharing scheme in order to reduce substantially the computational cost of the classifier, an essential issue for real-time applications. To this end, we compute a small set of R fern-features parameters, $\Omega = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_R\}$, that allows to compute multiple ferns at different locations but sharing the same random pairs of histogram bin indices [51]. Fig. 3-d shows a clarifying example where two random ferns with the same fern-features parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$ are computed at different image positions $\mathbf{u}$ and $\mathbf{v}$.

This scheme prevents computing a large number of ferns with specific parameters, which results in an increased computational cost. The set Ω is computed at random and keeps fixed during learning, see Fig. 4-b.

3.2.2 Online Boosting

We compute the WiLFs classifier using the online boosting algorithm proposed in [15, 16]. Using the same terminology as in this work, we define a selector $h^{sel}(\mathbf{x})$ to pick the weak classifier $h(\mathbf{x})$, from a shared pool of weak classifiers $\Psi = \{h_1, \ldots, h_K\}$, that best discriminates the target class $\mathcal{C}$ from the background one $\mathcal{B}$. Fig. 4 shows an illustrative schematic of the online boosting algorithm. These weak selectors are combined linearly to compute the two-class classifier $H(\mathbf{x})$:

$$H(\mathbf{x}) = \begin{cases} +1 \text{ if } \mathrm{conf}(\mathbf{x}) > \beta \\ -1 \text{ otherwise,} \end{cases} \tag{3}$$

$$\mathrm{conf}(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} h_t^{sel}(\mathbf{x}), \tag{4}$$

where $\mathrm{conf}(\mathbf{x})$ is the confidence of the classifier on predicting that $\mathbf{x}$ belongs to the class $\mathcal{C}$, T is the number of selectors, $\frac{1}{T}$ is a normalization factor, and β is a confidence threshold whose default value is 0.5. Thus, if the output of the classifier for a sample $\mathbf{x}$ is $H(\mathbf{x}) = +1$, the sample is assigned to the target or positive class. Otherwise, it is assigned to the background or negative class. This online classifier is computed initially at random, but it is progressively enhanced by updating the selectors as new samples become available.

In this work, we build the pool Ψ of K weak classifiers by computing ferns densely over the window $\mathbf{x}$, but using recursively the shared set of fern-features parameters Ω (observe Fig. 4-c). That is, we define each weak classifier $h_k \in \Psi$ by the probability that the sample $\mathbf{x}$ belongs to the target class using a fern $\mathbf{f}(\mathbf{x}; \mathbf{u}_k, \boldsymbol{\theta}_k)$ evaluated at position $\mathbf{u}_k \in \{\mathbf{u}_1, \ldots, \mathbf{u}_L\}$ and with features parameters $\boldsymbol{\theta}_k \in \Omega$. This can be expressed as

$$h_k(\mathbf{x}) = p(y = +1 | \mathbf{f}(\mathbf{x}; \mathbf{u}_k, \boldsymbol{\theta}_k) = z), \tag{5}$$

where $y = \{+1, -1\}$ is the class label, z is the fern output, and $\{\mathbf{u}_1, \ldots, \mathbf{u}_L\}$ are all possible locations in the image window $\mathbf{x}$. Since these probabilities follow a Bernoulli distribution $p(y | \mathbf{f}(\mathbf{x}; \mathbf{u}_k, \boldsymbol{\theta}_k) = z) \sim \mathrm{Ber}(y | \Theta_{k,z})$, we can write that

$$p(y = +1 | \mathbf{f}(\mathbf{x}; \mathbf{u}_k, \boldsymbol{\theta}_k) = z) \sim \Theta_{k,z}, \tag{6}$$

where $\Theta_{k,z}$ is the distribution parameter indicating the probability that a sample $\mathbf{x}$ in the fern $\mathbf{f}(\mathbf{x}; \mathbf{u}_k, \boldsymbol{\theta}_k)$ with output z belongs to the positive class. These parameters are computed online during learning through a Maximum Likelihood Estimate (MLE) over the labeled set of samples we have previously observed,

$$\Theta_{k,z} = \frac{\eta_{k,z}^{+1}}{\eta_{k,z}^{+1} + \eta_{k,z}^{-1}}, \tag{7}$$

where $\eta_{k,z}^{+1}$ and $\eta_{k,z}^{-1}$ are the accumulated numbers of positive and negative samples for each weak classifier h_k with fern output z. These numbers are initialized equally, but with new evidence they are progressively re-calculated according to the input samples $\langle \mathbf{x}, y \rangle$.

Next, and similarly to [15], each selector $h_t^{sel}(\mathbf{x})$ chooses the weak classifier $h_k(\mathbf{x}) \in \Psi$ that minimizes the misclassification error $e_{t,k}$,

$$e_{t,k} = \frac{\lambda_{t,k}^{wrong}}{\lambda_{t,k}^{wrong} + \lambda_{t,k}^{corr}}, \tag{8}$$

where $\lambda_{t,k}^{corr}$ and $\lambda_{t,k}^{wrong}$ are the weights of correctly and wrongly classified samples by $h_k(\mathbf{x})$ at the selector t. These

Algorithm 1: Online Classifier

Input: Previous online classifier $H(\mathbf{x})$
Input: Input sample $\langle \mathbf{x}, y \rangle$, with $y = \{+1, -1\}$
Input: Classification weights $\lambda_{t,k}^{corr}$, $\lambda_{t,k}^{wrong}$
Output: Updated online classifier $H(\mathbf{x})$

1 Initialize the importance weight $\lambda = 1$
2 **for** $r = 1, \ldots, R$ **do**
3 **for** $l = 1, \ldots, L$ **do**
4 Test the fern $\mathbf{f}(\mathbf{x}; \mathbf{u}_l, \boldsymbol{\theta}_r)$ for every location $\mathbf{u}_l$ of the sample $\mathbf{x}$ to compute the fern outputs $\mathbf{z}$. (Eq. 1 and 2)
5 **for** $k = 1, \ldots, K$ **do**
6 Update the estimate $\Theta_{k,z}$ of the weak classifier $h_k \in \Psi$ using the fern output z from previous computed outputs $\mathbf{z}$. (Eq. 7)
7 **if** $y = +1$ **then**
8 $\eta_{k,z}^{+1} = \eta_{k,z}^{+1} + 1$
9 **else**
10 $\eta_{k,z}^{-1} = \eta_{k,z}^{-1} + 1$
11 **for** $t = 1, \ldots, T$ **do**
12 **for** $k = 1, \ldots, K$ **do**
13 Update the correct and misclassified sample weights.
14 **if** $sign(h_k(\mathbf{x}) - \beta) = y$ **then**
15 $\lambda_{t,k}^{corr} = \lambda_{t,k}^{corr} + \lambda$
16 **else**
17 $\lambda_{t,k}^{wrong} = \lambda_{t,k}^{wrong} + \lambda$
18 Update the misclassification error $e_{t,k}$. (Eq. 8)
19 Select the weak classifier $h_t^{sel}(\mathbf{x}) = h_k(\mathbf{x})$ such that $h_k(\mathbf{x})$ minimizes the misclassification error $e_{t,k}$.
20 Update the importance weight λ,
21 **if** $sign(h_t^{sel}(\mathbf{x}) - \beta) = y$ **then**
22 $\lambda = \lambda \cdot \frac{1}{2 \cdot (1 - e_t)}$
23 **else**
24 $\lambda = \lambda \cdot \frac{1}{2 \cdot e_t}$
25 Assemble the final strong classifier $H(\mathbf{x}) = sign\left(\frac{1}{T}\sum_{t=1}^{T} h_t^{sel}(\mathbf{x}) - \beta\right)$. (Eq. 3 and 4)

classification weights are estimated and updated incrementally using the importance sample weight λ. Once the selector $h_t^{sel}(\mathbf{x})$ has selected a weak classifier, the importance weight λ is updated and given to the next selector $t + 1$ (see Fig. 4-d).

Algorithm 1 summarizes the computation of the online classifier. We refer the reader to lines 2 to 10 to observe the initial steps for testing and updating the set of weak classifiers Ψ. These steps are done efficiently since the small set of R fern-features parameters Ω is used to compute all K weak classifiers ($R << K$). Hence, the ferns can be computed on the sample $\mathbf{x}$ (lines 2 to 4) in advance to updating the weak classifiers (lines 5 to 10) and the selectors (lines 11 to 24). This reduces significantly the cost of the classifier.

3.3 Efficient Interactive Learning Approach

The proposed interactive learning approach is shown in Fig. 2. For each input image I and each window sample $\mathbf{x}$ (using a sliding window at multiple scales) the classifier $H(\mathbf{x})$ is initially tested on $\mathbf{x}$ to estimate its class label $y = \{+1, -1\}$ (Eq. 3 and Fig. 2-d). This results in performing face detection or automatic labeling[8]. Yet, in those situations where the classifier is ambivalent about its predictions (Fig. 2-f), the system uses the interactive human-machine learning to relabel the sample and update the classifier (Fig. 2-e).

In the following, we describe the proposed interactive learning approach to compute the classifier on the fly using minimal human supervision. Specifically, this approach is a pipeline consisting of three main stages: 1) an uncertainty-based active learning strategy to compute the classifier with highly informative samples (hard samples), that leads to a more discriminative classifier and a considerable reduction of human annotations; 2) an adaptive threshold to adjust the uncertainty region in function of the classifier performance, allowing to reduce the percentage of human assistance over time; and 3) a memory-based learning module that uses exemplar samples seen before to determine the class labels of difficult samples in advance to human intervention.

3.3.1 Active Learning

In this work, we resort to uncertainty-based active learning [30,46] to avoid *drifting*[3] problems, frequent in non- and semi-supervised learning approaches [15], and to decrease the degree of human interplay because the classifier is learned only with difficult samples falling within an uncertainty region, and which are labeled by the human user (like an oracle). In Fig. 5-a is shown a demo example that includes two samples classes. Positive samples are shown by circles while negative samples are indicated by squares. The uncertainty region (dashed lines) is defined by a threshold δ around the decision boundary (solid line) that, in turn, is defined by $\text{conf}(\mathbf{x}) = \beta$ (refer to Eq. 3). Note that only a small portion of samples (cyan and red samples) are labeled by humans and used to compute the online classifier. The remaining samples (gray samples) are unused[9] because they are redundant and do not provide any benefit to the classifier performance (shown in the experiments section). As a result, the overall number of human annotations decreases considerably.

As active learning algorithm, we opt for query-by-boosting [2] because it uses the confidence $\text{conf}(\mathbf{x})$ of the online boosted classifier $H(\mathbf{x})$ to determine whether an input sample $\mathbf{x}$ requires human annotation (see Fig. 2-f). This request $\mathcal{Q}_a$ can be written as:

$$\mathcal{Q}_a(\mathbf{x}) = \mathbb{I}(\beta + \delta/2 > \text{conf}(\mathbf{x}) > \beta - \delta/2), \tag{9}$$

where δ is the uncertainty threshold. If $\mathcal{Q}_a(\mathbf{x})$ is true the system asks for human assistance to get the sample class

[8] We use these terms interchangeably to express that the system discovers new face instances on images.

[9] Albeit the classifier is able to predict their class labels.

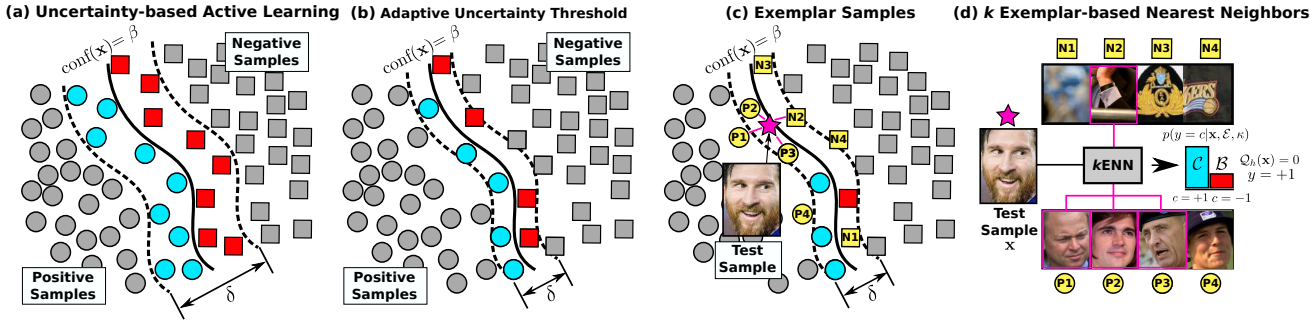

Fig. 5 Interactive learning approach. (a) Uncertainty-based active learning. (b) Adaptive uncertainty threshold δ. (c) Exemplar samples for predicting the class label of the test sample. (d) Memory-based learning using κENN. The test sample is assigned to the positive class ($y = +1$).

label (y) with which to update the classifier. On the other hand, if $\mathcal{Q}_a(\mathbf{x})$ is false the sample is discarded to compute the classifier. Note that in Fig. 5-a, $\mathcal{Q}_a(\mathbf{x})$ is true for all colored samples in the uncertainty region, whereas $\mathcal{Q}_a(\mathbf{x})$ is false for the rest.

3.3.2 Adaptive Uncertainty Threshold

In order to adapt the human assistance according to the performance of the classifier, as well as to diminish the degree of human supervision along the learning, we define an adaptive threshold that depends on the incremental classification rate over the requested samples. That can be formulated as,

$$\delta = 1 - \xi\phi, \tag{10}$$

where ξ is a sensitivity parameter assigned by the user, and ϕ measures the performance of the classifier. In turn, this performance rate can be computed by

$$\phi = N^c/N^q \tag{11}$$

being N^q and N^c the numbers of requested samples and correctly classified samples, respectively. A sample is said to be correctly classified when the class label y_c coming from the classifier agrees with the true class label given by the human user ($y_c = y_h$).

Fig. 5-b shows the adaptation of the uncertainty threshold δ. We see that the uncertainty region is narrowed resulting in a smaller number of human annotations (colored samples). This plays a similar role to stopping techniques [46].

3.3.3 κ Exemplar-based Nearest Neighbors

In order to reduce even more the level of human assistance, we propose to use in advance a memory-based classification method, that together with active learning, allows labeling new hard samples using a set of exemplar samples. Fig. 5-c shows some of them (yellow samples), which correspond to difficult samples observed before during learning.

In particular, we use κ Exemplar-based Nearest Neighbors (κENN) [33] that, in contrast to more conventional κNN techniques [37], the classification is done efficiently by comparing the test sample $\mathbf{x}$ against a small group of exemplar samples $\mathcal{E} = \{\mathbf{x}_1^e, \ldots, \mathbf{x}_E^e\}$. This classification can be written by

$$y = \arg\max_c p(y = c|\mathbf{x}, \mathcal{E}, \kappa), \quad c = \{+1, -1\} \tag{12}$$

where $p(y = c|\mathbf{x}, \mathcal{E}, \kappa)$ denotes the posterior probability of the sample $\mathbf{x}$ given the κ nearest samples in $\mathcal{E}$. Similarly, this posterior is calculated as

$$p(y = c|\mathbf{x}, \mathcal{E}, \kappa) = \sum_{i \in \mathcal{N}(\mathbf{x}, \kappa, \mathcal{E})} \omega_i \, \mathbb{I}(y(\mathbf{x}_i^e) = c) \tag{13}$$

where $\mathcal{N}(\mathbf{x}, \kappa, \mathcal{E})$ are the indices of the nearest exemplar samples, and $\omega_i = \frac{1}{d(\mathbf{x}, \mathbf{x}_i^e)}$ is the weight associated to the distance between samples $\mathbf{x}$ and $\mathbf{x}_i^e$. This distance is, in turn, defined as $d(\mathbf{x}, \mathbf{x}_i^e) = ||D(\mathbf{x}) - D(\mathbf{x}_i^e)||$, being $D(\mathbf{x}_j)$ the feature descriptor for the sample $\mathbf{x}_j$.

Since κENN is done with uncertain image samples inside the uncertainty region, we resort to a pre-trained Convolutional Neural Network (CNN) to compute high-level representations of these samples with which to perform nearest neighbors. For this goal, we use the VGG-16 network trained previously on the Imagenet database[6]. Specifically, each sample $\mathbf{x}$ is passed through the network to obtain a discriminative feature descriptor $D(\mathbf{x})$. In this work, we use the third from last layer as feature encoding.

As can be seen in Fig. 5-c, not all samples falling in the uncertainty area are considered exemplar samples. In order to control the size of the exemplar set $\mathcal{E}$ and thereby maintain efficiency, we introduce a scatter measure to select only those samples which are distant from other ones as exemplars. Then, a sample $\mathbf{x}$ is considered exemplar if it satisfies $d(\mathbf{x}, \mathbf{x}_i^e) > \zeta$ for all exemplars in $\mathcal{E}$, being ζ a distance threshold.

Fig. 5-d shows that the test sample is assigned correctly to the positive class ($y = +1$) using the class estimate y_m provided by the machine (κENN). This proves that difficult samples can be labeled automatically using past samples without human intervention.

Algorithm 2: Interactive Learning

Input: Online classifier $H(\mathbf{x})$
Input: Input sample $\mathbf{x}$
Input: Sensitivity parameter ξ
Input: Numbers of learning samples N^c and N^q
Input: Uncertainty classification threshold δ
Output: Updated classifier $H(\mathbf{x})$
Output: Updated numbers of learning samples N^c and N^q
Output: Updated uncertainty classification threshold δ

Test the classifier $H(\mathbf{x})$ in $\mathbf{x}$ to obtain the confidence $\text{conf}(\mathbf{x})$ and the class sample estimate $y = y_c$, being $y_c = H(\mathbf{x})$. (Eq. 3 and 4)
Assess if $\mathbf{x}$ falls in the uncertainty region defined by δ. (Eq. 9)
if $\mathcal{Q}_a(\mathbf{x}) = 1$ **then**
    Request machine assistance to predict the class sample label $y = y_m$, being y_m the output of κENN. (Eq. 12)
    Assess if $\mathbf{x}$ requires human assistance. (Eq. 14)
    if $\mathcal{Q}_h(\mathbf{x}) = 1$ **then**
        Request human assistance to provide the true class label $y = y_h$, being y_h the human annotation.
        Update the number of requested samples $N^q = N^q + 1$
        Update the number of correctly classified samples $N^c = N^c + \mathbb{I}(y_h = y_c)$
        Update the classifier performance $\phi = N^c/N^q$. (Eq. 11)
        Update the uncertainty threshold $\delta = 1 - \xi\phi$. (Eq. 10)
        Add the sample $\mathbf{x}$ in the exemplar set $\mathcal{E}$ if $\mathbf{x}$ is distant from other exemplar samples ($d(\mathbf{x}, \mathbf{x}^e_{i=1:E}) > \zeta$).
    Update the classifier $H(\mathbf{x})$ using the tuple $\langle \mathbf{x}, y \rangle$. (Alg. 1)

However, there are cases where κENN is not completely sure about its decision y_m, and the system ultimately requires the human assistance to label this sample ($y = y_h$). Similarly to Eq. 9, we define a second request $\mathcal{Q}_h$ for human intervention as:

$$\mathcal{Q}_h(\mathbf{x}) = \mathbb{I}\left(\mathcal{H}(p(y|\mathbf{x}, \mathcal{E}, \kappa)) > \rho\right) \tag{14}$$

where $\mathcal{H}$ is the Shannon entropy (used to measure to the confidence of κENN), and $\rho = 0.9$ is the default threshold for human assistance. If $\mathcal{Q}_h(x)$ is true the class sample label is given by the user (Fig. 2-k). Otherwise, the sample is annotated by κENN (Fig. 2-j).

Algorithm 2 shows the proposed interactive learning approach to compute the online classifier. Note that the human assistance is the last stage in the proposed pipeline, computing hence the classifier with small human supervision.

4 2D Classification Problem

In this section, we conduct a variety of synthetic experiments in order to observe and analyze in depth the main contributions of the proposed approach (see Fig. 6). For this purpose, we resort to 2D classification problems which have been widely used in the literature to validate and examine different methods [7,24], since they allow showing more clearly the effect of diverse parameters over the classification results and efficiency.

The first part of this section is devoted specifically to evaluate and compare WiLFs against other standard and related methods via supervised learning. By contrast, the second part focuses on evaluating the interactive human-machine learning approach, and compare it with other conventional learning strategies including supervised and active learning.

Since 2D classification is a simpler problem than face detection, instead of the proposed CNN-based descriptor, we opt for a binary feature descriptor and the Hamming distance to carry out κENN. This descriptor is computed as the output of a set of random binary features (2D decision stumps). For this work, the descriptor has 500 binary features.

4.1 Classifier Performance Evaluation

The performance evaluation of the presented classifier is carried out over two complex classification scenarios, observe Fig. 6, where each scenario consists of two sets of 2D samples belonging to either a positive or a negative class under a special distribution. Positive samples are indicated in the figure by cyan crosses while negative samples are shown by red circles. Both classes have 1000 random samples drawn from the class distributions. In the first scenario, the samples are distributed in different clusters with the goal of considering classification problems with multiple modalities, whereas the second scenario is a two-arm spiral used to represent complex and nonlinearly separable distributions.

The performance of WiLFs is evaluated and compared with other standard classifiers, namely Online Random Ferns (ORFs) [39], Random Forests (RForest) [7], GentleBoost classifier (GBoost) [12], and its offline counterpart BRFs [50]. The evaluation is done in terms of the most relevant parameters concerned with the computation the WiLFs classifier, such as the number of selectors (weak classifiers) T and the depth of ferns, which corresponds with the quantity of binary features per fern M. Importantly, all classifiers are computed via fully-supervised learning. The impact of other learning approaches are addressed in the next section.

For 2D classification scenarios, the classifiers are built using axis-aligned split functions (2D decision stumps) as binary features. Each decision stump f maps a given sample $\mathbf{x} \in [0,1] \times [0,1]$ to a Boolean label, $f(\mathbf{x}) = \mathbb{I}(x_j > \tau)$, where x_j is to a specific (horizontal or vertical) coordinate of $\mathbf{x}$, and τ is a random threshold in the interval $[0,1]$.

To build the fern-based classifiers, we use a large pool of 500 ferns (sets of decision stumps) computed at random. Every approach then uses a specific strategy to select and ensemble ferns into a set of weak classifiers and obtain the final classification rule $H(\mathbf{x})$. WilFs use an online boosting, BRFs make use of Real AdaBoost and ORFs simply choose the ferns randomly. Contrary, RForest picks features that maximize information gain at node-level to build the binary decision trees, and GBoost ensembles multiple one-dimensional decision stumps using GentleBoost.

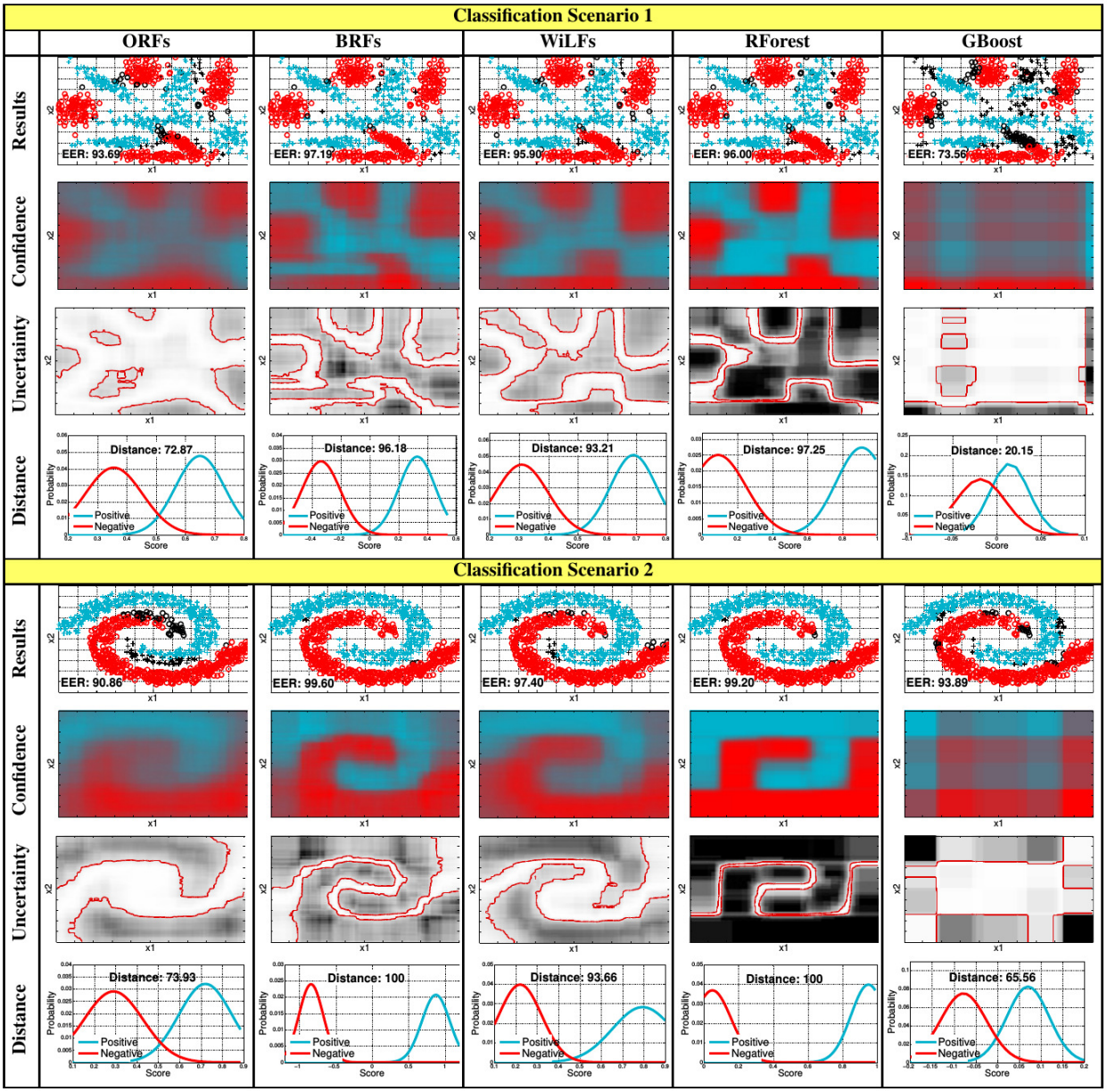

Fig. 6 2D classification results of WiLFs vs. ORFs [39], BRFs [50], Random Forest [7] and the GentleBoost classifier [12]. *Results rows*: classification results on 1000 positive and negative testing samples. Correctly classified samples are shown in cyan (positive samples) and red (negative samples). Misclassified samples are shown in black. *Confidence rows*: confidence maps provided by the classifiers over the entire 2D feature space. *Uncertainty rows*: uncertainty maps given by the classifier where brighter regions denote uncertain classification values. Red contours indicate uncertainty of 90%. *Distance rows*: score distributions distances between the positive and negative classes.

All methods were tested on samples drawn from the same distributions as those used in the training sets, and each experiment was repeated 10 times to account for randomness in feature selection. We base our analysis on two metrics: the Break-Even Point (BEP) on the precision-recall curve[10]; and the Hellinger distance[11] that measures the degree of separability between the positive and negative distributions, observe Table 1. We also report the performance with regard to the training and testing (run) times for various numbers of weak classifiers T and tree depth values D, that in the case of WiLFs, BRFs and ORFs corresponds to the number of binary features M. For the GBoost, we directly use an implementation publicly available[12]. Since no trees are considered, depth $D = 1$.

A visual comparison of the classification performance of all presented methods is shown in Fig. 6, for both classi-

[10] BEP is the point in the curve where precision=recall.

[11] The squared Hellinger distance for two distributions P and Q is defined as: $H^2(P,Q) = 1 - \sqrt{k_1/k_2}\exp(-0.25k_3/k_2)$, with $k_1 = 2\sigma_P\sigma_Q$, $k_2 = \sigma_P^2 + \sigma_Q^2$, and $k_3 = (\mu_P - \mu_Q)^2$.

[12] http:/people.csail.mit.edu/torralba/shortCourseRLOC/boosting/boosting.html

	Break-Even Point [%]				Hellinger Distance [%]				Training Time [sec.]				Run Times [msec.]				D
Classification Scenario 1																	
ORFs	58.5	63.5	69.1	70.3	7.1	9.2	11.2	16.2	0.03	0.03	0.04	0.05	0.02	0.02	0.02	0.02	1
	65.4	71.8	75.9	78.8	15.2	20.0	22.0	27.2	0.03	0.04	0.04	0.05	0.02	0.02	0.02	0.02	2
	81.2	85.8	89.9	91.7	30.8	43.4	56.0	61.1	0.04	0.04	0.05	0.06	0.02	0.02	0.03	0.03	5
	91.1	92.7	94.2	95.3	61.6	66.8	76.3	81.0	0.04	0.04	0.06	0.08	0.02	0.02	0.03	0.05	8
BRFs	73.6	73.2	74.5	76.0	17.5	20.0	23.0	21.5	0.15	0.27	0.60	1.17	0.02	0.02	0.02	0.02	1
	92.1	95.4	96.9	97.4	43.6	68.2	77.8	81.9	0.17	0.28	0.62	1.18	0.02	0.02	0.02	0.02	2
	95.9	97.1	96.9	97.6	81.3	87.0	89.1	92.5	0.27	0.38	0.73	1.29	0.02	0.02	0.03	0.03	5
	97.0	97.3	97.1	97.5	89.0	92.4	93.8	95.5	0.39	0.51	0.88	1.49	0.02	0.02	0.03	0.05	8
WiLFs	76.1	71.1	75.8	72.1	16.4	18.8	13.2	17.9	1.03	1.57	3.23	6.00	0.02	0.02	0.02	0.02	1
	75.5	80.1	84.6	83.9	24.3	30.0	26.7	31.9	1.04	1.61	3.29	6.08	0.02	0.02	0.02	0.02	2
	93.5	95.0	95.6	96.6	65.0	76.5	73.9	78.7	1.16	1.74	3.44	6.24	0.02	0.02	0.03	0.03	5
	95.0	96.0	96.8	96.8	75.9	84.3	85.8	89.1	1.31	1.92	3.65	6.49	0.02	0.02	0.03	0.05	8
RForest	74.7	76.3	74.2	74.7	33.2	38.2	43.9	38.8	0.08	0.14	0.34	0.70	0.13	0.23	0.54	1.06	1
	68.1	68.5	75.7	74.9	25.7	29.0	30.3	29.0	0.19	0.38	0.94	1.87	0.18	0.33	0.79	1.55	2
	91.9	95.1	95.7	95.9	66.4	74.4	80.1	82.4	0.78	1.58	3.95	7.88	0.30	0.57	1.40	2.74	5
	96.8	97.0	97.0	97.2	97.4	97.6	98.0	98.5	1.56	3.20	7.77	15.65	0.35	0.69	1.73	3.34	8
GBoost	73.4	73.1	73.7	76.0	18.7	21.6	23.5	22.3	0.00	0.00	0.01	0.02	0.02	0.02	0.02	0.02	1
Classification Scenario 2																	
ORFs	72.8	75.9	75.0	73.4	26.2	30.2	29.4	30.2	0.04	0.04	0.04	0.05	0.02	0.02	0.02	0.02	1
	76.5	76.4	76.9	75.7	30.4	33.0	34.3	34.9	0.04	0.04	0.04	0.05	0.02	0.02	0.02	0.02	2
	82.4	82.6	82.9	82.2	45.4	46.8	52.2	52.0	0.04	0.04	0.05	0.06	0.02	0.02	0.03	0.03	5
	88.5	88.7	90.3	89.7	62.5	66.6	73.1	73.3	0.04	0.04	0.06	0.08	0.02	0.02	0.03	0.05	8
BRFs	90.8	94.9	94.9	94.3	51.4	56.2	60.6	61.3	0.16	0.27	0.61	1.15	0.02	0.02	0.02	0.02	1
	94.4	98.2	98.9	99.6	67.7	80.9	88.2	86.4	0.17	0.28	0.62	1.17	0.02	0.02	0.02	0.02	2
	98.4	99.6	99.7	99.8	93.2	97.4	98.2	98.3	0.27	0.39	0.72	1.28	0.02	0.02	0.03	0.03	5
	99.1	99.5	99.8	99.8	98.9	99.8	100	100	0.40	0.52	0.88	1.48	0.02	0.02	0.03	0.05	8
WiLFs	84.7	86.8	87.8	85.4	37.9	41.1	43.0	42.6	1.03	1.52	3.14	5.61	0.02	0.02	0.02	0.02	1
	87.6	88.9	90.0	90.4	41.9	42.1	44.0	41.6	1.02	1.55	3.08	5.62	0.02	0.02	0.02	0.02	2
	90.3	94.4	95.0	95.1	60.6	66.9	69.4	67.9	1.14	1.70	3.19	5.77	0.02	0.02	0.03	0.03	5
	97.7	97.6	98.1	98.4	88.0	87.6	90.6	92.6	1.30	1.86	3.44	6.05	0.02	0.02	0.03	0.05	8
RForest	78.6	79.3	77.5	74.8	36.5	35.1	36.1	36.0	0.08	0.14	0.40	0.69	0.14	0.26	0.58	1.06	1
	79.8	78.2	78.4	77.2	33.6	32.9	33.0	33.7	0.20	0.39	0.93	1.84	0.19	0.35	0.76	1.49	2
	92.8	94.7	94.7	94.7	83.5	90.1	87.7	90.7	0.69	1.51	3.29	6.97	0.28	0.53	1.19	2.35	5
	98.1	98.2	98.6	98.8	99.8	99.5	100	100	1.33	2.50	5.96	11.53	0.33	0.61	1.41	2.71	8
GBoost	88.7	90.8	93.8	94.4	47.0	56.0	59.6	61.4	0.00	0.00	0.01	0.02	0.02	0.02	0.02	0.02	1
T	5	10	25	50	5	10	25	50	5	10	25	50	5	10	25	50	

Table 1 Classification performance of RFs, BRFs, WiLFs, RForest and GBoost in the scenario of Fig. 6 for different values of weak classifiers T and tree depth D. First column: mean values of Break-Even Points (BEP) on the precision-recall curve. Second column: Hellinger distances between classes. Third and fourth columns: computational times for training and testing the classifiers.

fication scenarios and for a case with $T = 50$ and $D = 8$ ($D = 1$ for GBoost). Observe that BRFs and RForest obtain the best classification performance, followed closely by the WiLFs classifier. These methods attain higher classification rates (BEP) and larger separability between classes than the RFs and GBoost classifiers. Note also that WiLFs, BRFs and RForest clearly define decision boundaries in the confidence and uncertainty maps whereas RFs and GBoost yield washed-out maps, indicating larger risk of misclassification.

BRFs obtain better classification results than WiLFs because BRFs are trained with all the samples in every iteration of the algorithm. This yields a better decision hypothesis. Conversely, WiLFs are updated with one input sample at time, what results in a less discriminative classifier since the weak classifiers are adjusted mainly with this single sample and past evidence.

Table 1 shows an exhaustive quantitative analysis of the performance and efficiency of the above methods according to different classifier configurations, such as the number of weak classifiers. Once again, BRFs and RForest achieve the best performance rates (BEP and Hellinger distances). WiLFs obtain also competitive results with the benefit of being computed online and available anytime. The ORFs classifier also obtains remarkable classification rates but using a more stringent configuration (large values of weak learners and fern depth). By contrast, GBoost yields a moderate performance since this classifier just combines individually decision stumps. Thus, this classifier cannot cope with complex and nonlinearly separable distributions.

Training and run times[13] for the diverse methods are also shown in Table 1. We see that the fern-based methods and the GBoost classifier are fast in run times. Conversely, RForest presents a much higher computational cost when the values of T and D get larger. The same behavior occurs for training times where RForest is the most computationally expensive method. This is because the size of the trees grow exponentially with the depth D, requiring thus to compute much more features with an iterative process based on information gain. On the other hand, GBoost and ORFs are the fastest classifiers since GBoost uses single features and ORFs select the ferns randomly and keep them fixed during training (fern probabilities are updated only). The BRFs classifier shows a remarkable efficiency while attains the best classification rates. This classifier iterates over the pool

[13] Training times refer to the times spent on computing the classifier using all training samples, whereas run times are the times spent on testing the classifier on a test sample.

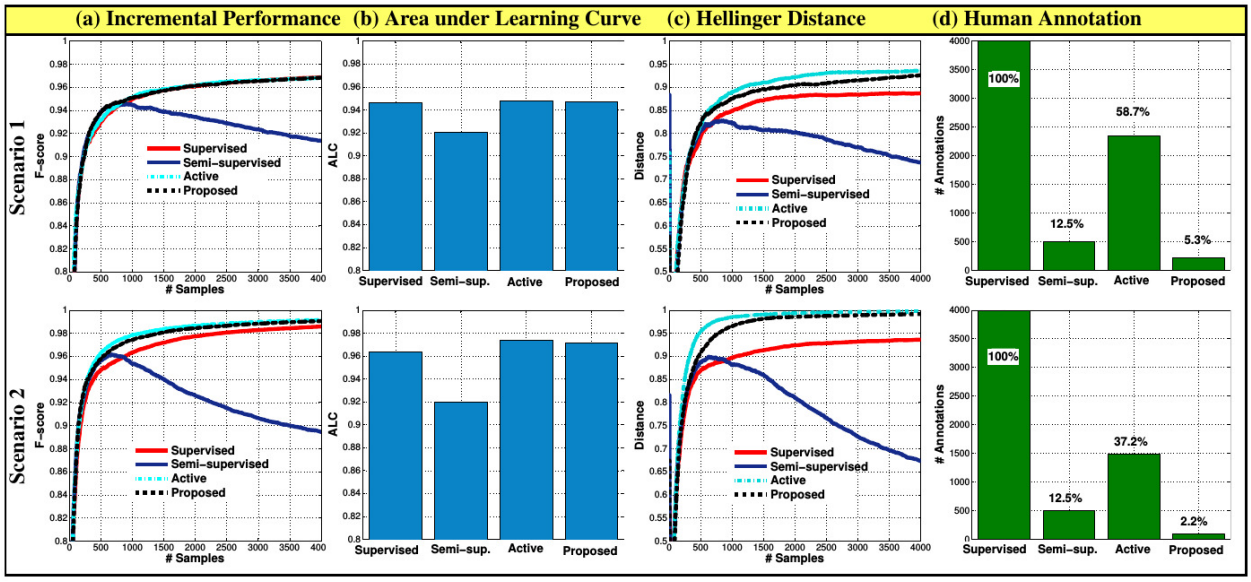

Fig. 7 Classification performance of the WiLFs classifier using four different learning strategies: supervised, semi-supervised, active and the proposed method. First column: Incremental learning curves of WiLFs using the F-measure. Second column: Areas under the learning curve (ALC). Third column: Distribution distances between the positive and negative classes along the learning step. Fourth column: Amounts of human annotations (human labels) used to compute the classifier for each learning scheme.

of ferns to choose the most discriminative ones. This procedure is also done in WiLFs but for every training sample because of this classifier is learned and updated incrementally. This slows down the training step significantly but allows having an online classifier that can be adapted to new and unexpected conditions, contrary to BRFs. Nevertheless, once the WiLFs classifier is computed, evaluating this classifier is fast because it does not involve any updating process. This is shown in the run times reported in Table 1 where WiLFs give identical times that ORFs and BRFs.

4.2 Classifier Learning Evaluation

Through this section we evaluate intensively the performance of WiLFs with the proposed interactive learning approach, and compare it against other conventional learning strategies. Specifically, we analyze the impact of each of the constituents of the proposed method on the classification rates and the amount of human assistance during the learning step.

Unlike the previous section where two separated sample sets were used to compute and test the classifiers (training and testing steps), so as to compare online and offline methods, this section evaluates the classification performance of WiLFs incrementally where the classifier is computed and tested at the same time. This proceeds as follows: given an input sample $\mathbf{x}$, the classifier $H(\mathbf{x})$ is first tested on this sample in order to estimate its class label y. Then, this class estimate is compared with the true sample label y to compute the incremental performance of the classifier[14]. Subsequently, the classifier is updated with the input sample according to the given learning approach. This procedure is done repeatedly for all input samples.

4.2.1 Learning Approaches Comparison

The classification results of WiLFs for both classification scenarios (including 2000 positive and negative samples) and four different learning approaches are shown in Fig. 7. These learning approaches are:

Supervised: The WiLFs classifier is computed with all $N = 4000$ samples and using human labels[15]. That is, for each sample $\mathbf{x}$ the human user provides the corresponding class label y.

Semi-supervised: The first n samples are labeled by the human (human labels), whereas the remaining ones are labeled using the classifier confidence (machine labels). The latter is computed for a sample $\mathbf{x}$ according to $y = H(\mathbf{x})$, see Eq. 3. In this work we use a value of $n = 500$.

Active: The classifier is only computed with samples with high uncertainty about their class predictions. The human resolves the ambiguity by providing the sample label. Here, we use a fixed uncertainty threshold of $\delta = 0.2$ and the assistance criterion defined in Eq. 9.

Proposed: The classifier uses active learning in combination with an adaptive uncertainty threshold δ (Eq. 10). Moreover, in cases of high uncertainty, the approach resorts first to the memory-based classification method (Eq. 12) to estimate the

[14] We assume knowing the true class labels of all samples (ground truth labels) for evaluation purposes.

[15] For these 2D experiments, we assume that the human labels correspond to the ground truth labels of the class distributions.

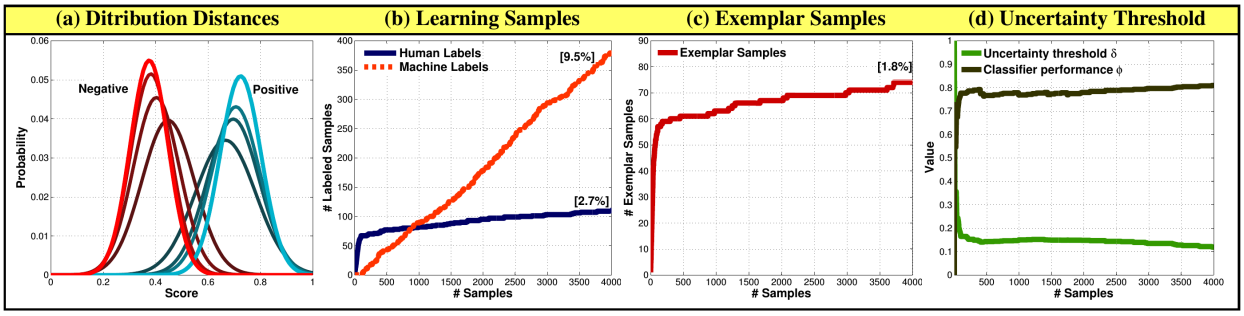

Fig. 8 Incremental performance of the proposed learning approach for the classification scenario 2. First column: Positive and negative class distributions for diffirent learning instances. Brighter colors denote the last learning instances. Second column: Amounts and overall percentages of human and machine labels used to compute the WiLFs. Third column: Numbers and final percentage of used exemplar samples. Fourth column: Adaptive behavior of the uncertainty threshold along the learning process.

sample label. If this method still presents ambiguity in its decision, the approach finally resorts to the human assistance.

The incremental classification rates of WiLFs for the four learning approach are plotted in Fig. 7-a. We measure the classification performance using the F-score. Note that all learning approaches have the same tendency initially, they begin with very low values of classification and then start to improve substantially as more samples are available. The supervised, active and proposed learning methods achieve the highest classification scores, contrary to the semi-supervised approach. This is also seen in Fig. 7-b where the areas under these learning curves (ALC) are displayed. Once more, the active and the proposed approaches perform best, followed closely by the supervised learning. The semi-supervised learning, conversely, deteriorates the classifier performance. This is because the self-learning that is applied to this method suffers from drifting, making the classifier to be constantly updated with erroneously labeled samples. Specifically, this classifier deteriorates after $n = 500$ samples, which is the number of human labels in these experiments.

Fig. 7-c shows the two-class distribution distances through the learning step. Again, observe that all curves are monotonic increasing functions, except the semi-supervised method that decreases over time. The active and the proposed approach obtain the largest separability between classes, reducing hence the risk of misclassification. Interestingly, the supervised method produces much lower distances, despite using the full set of samples, than the active and the proposed learning methods. This occurs because these latter methods compute the classifier with difficult samples only, and because they use human-made labels to remove the drifting problem. This focuses the classifier mainly on the decision boundaries and makes it more discriminative that using all training samples (supervised method).

With respect to the degree of human supervision, Fig. 7-d shows the total numbers and percentages of samples labeled by the human to train the classifier. Notice that the supervised approach uses all samples with their corresponding human labels, while in the semi-supervised approach, the human just annotates the first 500 samples. The other samples are labeled by the classifier (machine labels). This reduces considerably the annotation cost but at the expense of reducing the classification rates, see Fig. 7-a,b. On the other hand, the active method achieves remarkable classification rates but with a much higher cost of labeling. By contrast, the proposed approach reduces significantly the number of human annotations by using a second classification stage based on exemplar samples, and using an adaptive uncertainty threshold that depends on the classifier confidence. On average, we reduce by 96% the human annotations in relation to the supervised method and 44% on the active method. This shows that the proposed approach obtains high classification rates with limited human-labeling cost.

4.2.2 Proposed Learning Performance

The progressive performance of the proposed method on the two-arm classification scenario is also shown in Fig. 8. For instance, the evolution of the positive and negative class distributions during the learning is illustrated in Fig. 8-a, where it is seen that the distributions are increasingly further apart because the classifier is more discriminative as more samples are used. In Fig. 8-b, we plot the incremental numbers and general percentages of samples used to compute the classifier according to whether they were labeled by either the human (human labels) or the same classifier (machine labels). As can be seen, the number of human-labeled samples is much smaller than the machine-labeled samples and represents only a very small portion of the total set of samples (2.7%), showing again that WiLFs can be trained with little human effort[16].

Furthermore, we see that over time the classifier is built mostly with machine labels that with human labels. This is due to the system already has a sufficient number of exemplar samples that allow determining the class of incoming

[16] Note that although the classifier is computed with both kinds of samples, the annotation cost uniquely corresponds to human labels, since machine labels are automatically processed.

samples with enough accuracy. This is observed in Fig. 8-c, where the number of exemplar samples is given. Initially, this number grows quickly because the system lacks of exemplar samples and it uses thus human labels, but once the system has achieved a sufficient quantity of these samples, the number of new exemplar samples is greatly reduced, showing a saturation tendency. Note also that the percentage of exemplar samples (1.8%) is less than the percentage of human-labeled samples, indicating that the quantity of exemplar samples does not necessarily correspond with the degree of human interventions.

The behavior of the uncertainty threshold δ along the learning is illustrated in Fig. 8-d. Since the threshold is adapted in accordance to the classifier performance ϕ, this threshold is decreased progressively over time because the classifier is gradually more discriminative.

5 Experiments

WiLFs are validated in this section in three datasets for face detection in the wild. They are the FDDB [21], GENKI [1], and AFW [59] datasets. FDDB contains 2845 images with 5171 faces collected from Yahoo news website, AFW contains 468 faces in 205 images, and GENKI has 3500 images with a wide range of faces collected from Internet. These datasets include faces under very difficult conditions such as blurring, out-of-plane rotations, large intra-class appearance and occlusions.

In the next experiments, unless otherwise stated, the classifier is computed with $T = 750$ weak classifiers, $R = 12$ random ferns, and $M = 8$ binary features. For the interactive learning, we use $\kappa = 11$ exemplar samples, a human assistance threshold of $\rho = 0.7$, and a learning rate of $\xi = 1.0$. The histogram of oriented gradients is built using four gradient channels and a cell size of 3×3 pixels. The size of input images are normalized by height to 640 pixels.

All experiments are conducted in Matlab using a CPU Intel i7 (2.20 GHz). To speed up some time-consuming processes, some functions were implemented using mex-files and OpenCV.

5.1 Feature Descriptor

Before evaluating WiLFs as a whole, we first test the performance of the feature descriptor to perform κENN. Particularly, we compare the proposed CNN-based descriptor[17] against other more conventional descriptors consisting of vectors of HOG, RGB, and gray-scale pixel values in the image samples. Table 2 shows these feature descriptors and their classification rates to perform nearest neighbors with a set of face and background samples (1000 image patches from FDDB) selected nearby the classification boundary (β=0.5). For these descriptors, the Euclidean metric is applied to measure the distance between two samples.

[17] http://www.vlfeat.org/matconvnet/pretrained/

Descriptor Evaluation			
Descriptor	**Recall**	**Precision**	**F-Score**
Euclidean Distance			
HOG	70.6	81.1	75.5
RGB	90.8	81.4	85.8
Gray	81.6	84.1	82.8
CNN	98.6	89.1	93.6
Hamming Distance			
BinHOG	99.8	62.4	76.8
BinRGB	96.2	80.3	87.5
BinGray	95.4	84.7	89.7

Table 2 Classification rates (%) according to different descriptors.

Classification and Annotation Rates					
	Human Assistance threshold (ρ)				
	0.4	0.6	0.8	0.9	1.0
Gray-Binary Descriptor (BinGray)					
F-Score	98.5	97.7	96.0	95.1	89.7
Annotation	56.5	35.3	23.0	17.1	0.0
Convolutional Neural Network (CNN)					
F-Score	99.4	98.4	97.3	97.1	93.6
Annotation	35.4	21.4	13.1	12.4	0.0

Table 3 Classification rates according to the assistance threshold ρ.

In addition, we compute binary descriptors on HOG, RGB, and Gray vectors to obtain more compact and faster descriptors using the Hamming distance [5,42]. In detail, these binary descriptors compute a set Boolean comparisons between different feature values. For example, the BinRGB descriptor compares pixel intensities for different color channels. In this experiment, 1000 binary features are chosen at random.

Looking at Table 2, the CNN-based descriptor attains the best classification scores followed by the BinGray descriptor. This result shows that the deep neural network can be used as feature encoder to obtain a discriminative and reliable representation to perform κENN. Despite the good results, we also see that the method is not perfect (F-Score of 93.6%) and that the precision rate is not so high (89.1%), indicating that false positives may appear during the learning and produce the drifting of the classifier. However, this is not a big issue since the most difficult samples are assigned for human assistance. For this experiment, the human intervention was discarded by setting the assistance threshold $\rho = 1$.

The classification rates for the BinGray and CNN-based descriptors in terms of the human assistance are given in Table 3. Notice that the classification score increases as the threshold ρ gets smaller. This is because as the assistance threshold is reduced, more hard samples are labeled by humans and less by κENN. However, this is at the expense of an increase in the annotation cost. Besides, observe that the CNN-based descriptor outperforms again to the BinGray descriptor, both in classification and annotation cost.

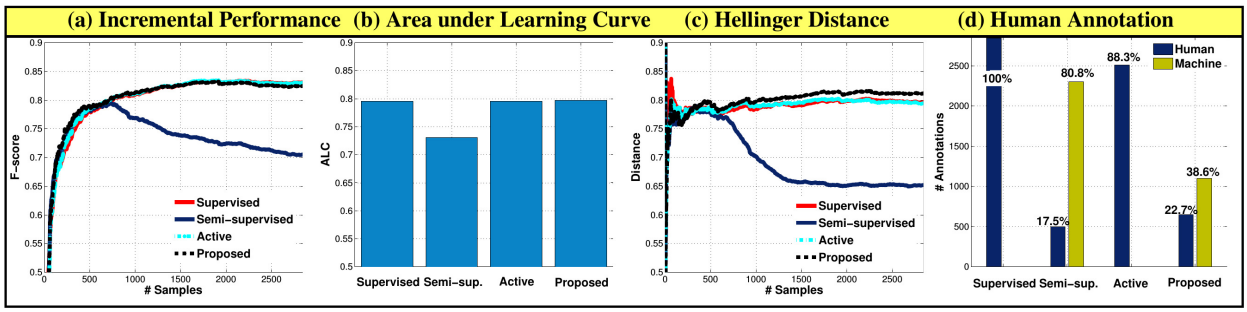

Fig. 9 Incremental detection rates of the proposed method for different learning approaches on the FDDB dataset.

5.2 Learning Approaches Comparison

Similarly to the 2D classification problem, WiLFs are evaluated in the FDDB dataset for different learning approaches. Each one with a particular degree of human supervision. They are: supervised, semi-supervised, active, and the proposed interactive learning using κENN. Fig. 9 shows the incremental learning and detection performance of WiLFs using these learning approaches.

In Fig. 9-a,b we see that all learning approaches produce pretty similar detection rates (ALC and F-Score), except the semi-supervised approach which deteriorates once the supervision step has finished (500 samples). This behavior is also seen in Fig. 9-c where the Hellinger distance for each approach is plotted.

Although all approaches provide similar classification scores, the great benefit of the proposed method is that the cost of human annotation during learning is significantly reduced. This is depicted in Fig. 9-d where the percentages of human and machine annotations are indicated. Here, it is worth mentioning that the human annotation is requested if any of the test windows in the input image is uncertain and falls within the uncertainty region (see Fig. 2 and Fig. 5). Then, it is very likely that most images are assisted because of the large number of windows inside them (about millions). This explains, in part, why the human annotation cost is relatively high in comparison with the 2D classification problem. The other reason is clearly that the face detection problem is much more complex than classifying samples in a two-dimensional space.

5.3 Online Face Detection

Some face detection results of WiLFs through the learning are shown in Fig. 1. The output of WiLFs is represented by colored rectangles. Green rectangles correspond to true positive detections, whereas red ones are false positives. Black boxes indicate the ground truth [21]. The letter H stands for human annotation, M for machine annotation, and X for exemplar samples. Hence, if an image contains any of these letters, this means that this image has been used for updating the classifier with human/machine annotations or used for extracting exemplar samples.

Fig. 10 Positive and negative exemplar samples extracted during the training of the WiLFs classifier. These samples are used to perform $\mathcal{K}$-Exemplar Nearest Neighbors ($\mathcal{K}$ENN).

Proposed Learning Approach			
Method	**ALC**	**Hellinger Distance**	**Human Annotation**
ACT	79.7	79.1	2494 [87.6%]
ACT+AUT	80.1	79.4	2112 [74.2%]
ACT+AUT+$\mathcal{K}$ENN	79.7	81.2	646 [22.7%]

Table 4 Classification results of WiLFs according to the main constituents of the proposed interactive learning approach: active learning (ACT), uncertainty threshold (AUT), and the κ Exemplar-based Nearest Neighbor ($\mathcal{K}$ENN).

Observe that the method is able to detect most faces using small human supervision. In early steps of learning, the human assistance is needed much often, but as the classifier gets more confident the human interplay decreases gradually giving way to machine annotation ($\mathcal{K}$ENN). Note also that some images are unused to compute the classifier because the faces in these images are relatively easy to classify and they do not fall in the uncertainty region. Moreover, we see that the exemplar samples are computed with human annotations so as to prevent false positives to be added as exemplar samples.

We show in Fig. 10 some positive and negative exemplar samples extracted during learning. They correspond to image windows with certain degree of difficulty. For example, the positive samples (first two rows) are faces having out-the-plane rotations and occlusions. On the other hand, negative samples correspond to background regions with some patterns resembling to facial components such as eyes.

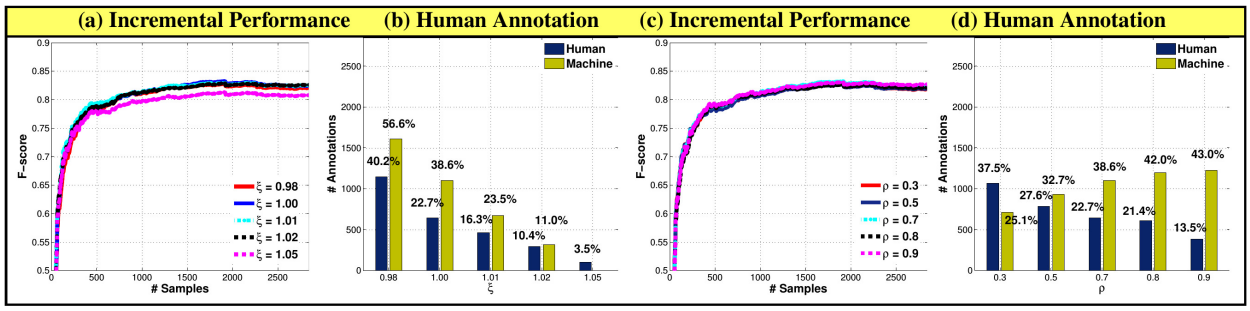

Fig. 11 Detection performance and annotation cost of WiLFs according to the sensitivity parameter ξ and the human assistance threshold ρ.

5.4 Interactive Learning Approach

The impact of each constituent of the proposed interactive learning approach is indicated in Table 4. Keep in mind that this approach consists of three main components: active learning (ACT), an adaptive uncertainty threshold (AUT), and a memory-based module based on κENN. Note that with each component the annotation cost is substantially decreased while the Hellinger distance and ALC present similar values. The full method obtains similar classification scores to other learning approaches using a quarter of face annotations. This is particularly significant in comparison with conventional supervised approaches which need all annotations to compute classifiers. The proposed approach can even further reduce the cost of human-made annotations by adjusting the sensitivity ξ and the assistance threshold ρ parameters. This will be evidenced in following sections.

5.5 Interactive Object Annotation

Another contribution of this work is that WiLFs can be used for interactively annotating face instances over a stream of images. This is a by-product of the proposed method that automatically discovers new instances as the classifier is computed and tested online. Particularly, WiLFs achieve a F-Score of 82.5% over the entire dataset, representing a recall of 71.6% and a precision of 97.2%, with an human annotation cost of 22.7%. This means that WiLFs annotate automatically 48.9% of faces, what is twice the quantity of annotations made by humans. In detail, this corresponds to 2528 faces given that the dataset includes 5171 faces.

5.6 Learning Parameters

In this section we see the effect of the learning parameters ξ and ρ over the detection performance and the amount of human labeling. Fig. 11 shows the detection score and human annotation cost for different parameters values.

We observe in Fig. 11-a,b the performance of WiLFs for varying values of the sensitivity parameter ξ. As this parameter gets larger, both the human and machine annotations are gradually reduced to the point that the classification score drops and the machine annotations disappear. This is visible for the case $\xi = 1.05$. For the other values the incremental detection plots present similar scores.

The choice of this parameter is, therefore, a trade off between the cost of labeling and the detection performance. With small values of ξ WiLFs use more training samples to compute the classifier and attain good detection rates. Conversely, high values of ξ reduce the uncertainty threshold δ to a greater extent, limiting the human interplay and training the classifier with insufficient number of samples. We suggest that a value of $\xi = 1$ is a good compromise. Nevertheless, we note that for a value of $\xi = 1.02$ the annotation cost is reduced by half whereas the F-Score remains the same.

With respect to the human assistance threshold ρ, the incremental detection performance of WiLFs is visualized in Fig. 11-c. For all the addressed values of ρ the proposed method shows very similar scores. This is because in all cases the classifier is computed approximately with the same quantity of training samples (around 60%). The only difference lies in whether the samples are labeled by humans or by κENN. This can be seen in Fig. 11-d where the number of human annotations is lessened a long with the increase of ρ, while the number of machine annotations is augmented. This behavior occurs because the parameter ρ controls the confidence of κENN to annotate automatically input samples. Therefore, if the value of ρ is small the learning system has less confidence on κENN and it requires then more human intervention.

We have found that for the CNN-based descriptor a value of ρ around 0.7 is a good trade off between remarkable detection rates and low annotation cost. With larger values the method uses more often the κENN system, increasing the risk of adding false positives. Keep in mind that κENN is not perfect as it was shown in previous sections. On the other hand, using small values of ρ imply more human assistance.

5.7 Number of Weak Classifiers

Table 5 shows the detection results of WiLFs in function of the number of weak classifiers (T) used to compute the online classifier. The table shows that increasing the number of weak classifiers leads to a more discriminative classifier which obtains better detection rates. Yet, we found that beyond 750 weak classifiers the method does not improve significantly the rates. It is interesting to observe that the cost of human annotation augments with the number of weak clas-

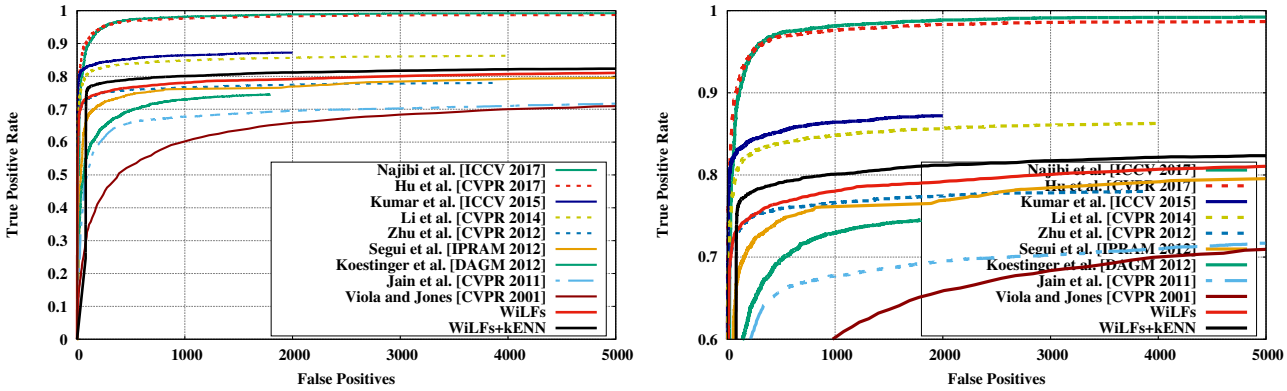

Fig. 12 Detection plots provided by WiLFs and other methods on the FDDB dataset. Left: ROC curves. Right: Zoomed curves.

Weak Classifiers			
# Weak Classifiers	**250**	**500**	**750**
Learning and Detection Rates (%)			
ALC	73.1	79.5	79.7
F-score	78.8	81.9	82.5
Hellinger Distance	80.9	80.4	81.2
Human Annotation	16.5	17.2	22.7
Computation Times [in seconds]			
Detection	0.31	0.31	0.30
Updating	0.28	0.40	0.62
Total	0.45	0.49	0.68

Table 5 Learning and detection rates and computation times of WiLFs in terms of the number of weak classifiers T on the FDDB dataset.

sifiers. This is presumably because of the larger number of weak hypotheses supporting the final classification.

Table 5 also indicates the average computational times, per image, of testing and updating the classifier during the online learning. The detection time remains constant despite having more weak classifiers. This is due to the feature sharing scheme and the use of an efficient classifier based on random ferns (Sec. 3.2). We would like to remark that this detection time is short given that the detection is done using a sliding window at all image locations and multiple scales.

On the other hand, we see that the time spent to update the classifier increases with the number of weak classifiers. This occurs because the online boosting algorithm iterates on T selectors in order to choose the most discriminative weak classifiers, observe Alg. 1. Although updating the classifier is the most computationally expensive process, this is done fewer times and only for those images requiring human assistance. In short, WiLFs run about 1 FPS to simultaneously learn and detect faces in images.

5.8 Comparison with the state of the art

In this section we evaluate and compare WiLFs against other face detection approaches on the FDDB [21], GENKI [1], and AFW [59] databases. Since the list of approaches is extensive, we only mention some of them in this work.

FDDB Dataset. Unlike the previous experiments, where WiLFs are sequentially learned and tested at the same time through the entire dataset, here we follow the evaluation procedure proposed in [21] using 10-fold cross-validation. This allows to compare our online method against other offline methods using separated sets of training and test images. Indeed, our method does not need this kind of validation since the classifier is tested and evaluated in advance to the updating step, measuring thus the generalization capability in an online way.

We show in Fig. 12 the detection plots for two versions of WiLFs. The first one corresponds to the standard WiLFs proposed and tested in previous experiments. The second one (WiLFs + κENN) is WiLFs tested at a larger image resolution (i.e height normalization of 1020 pixels instead of 640 pixels) in order to detect very small faces in images. This version also adds a verification step during detection using κENN. That is, once the classifier has generated a set of potential detections, κENN is used to filter out false detections in the uncertainty region. This differs from past experiments. Remember that the memory-based method (κENN) is used only to assist and update the online classifier but not for detection. Here, however, we see that combining OBRFs and κENN improves the detection rates at the expense of an increase of the computational cost.

Looking at WiLFs together other works in the state of the art, we notice that our detection results are good (especially for an online algorithm) but distant from other recent methods, see for instance Hu *et al.* [19] and Najibi *et al.* [38]. We refer mainly to deep convolutional networks that are trained for long periods of time. In our case WiLFs are a more straightforward method based on extremely randomized trees and boosting in order to have an online and efficient classifier. Nevertheless, we consider that introducing deep-learning ideas can increment significantly the performance of classical methods. That is our case where WiLFs make use of a deep network trained in advance to compute feature descriptors. Moreover, unlike deep networks that are computed offline using a fully supervised learning, WiLFs are trained incrementally using small human supervision.

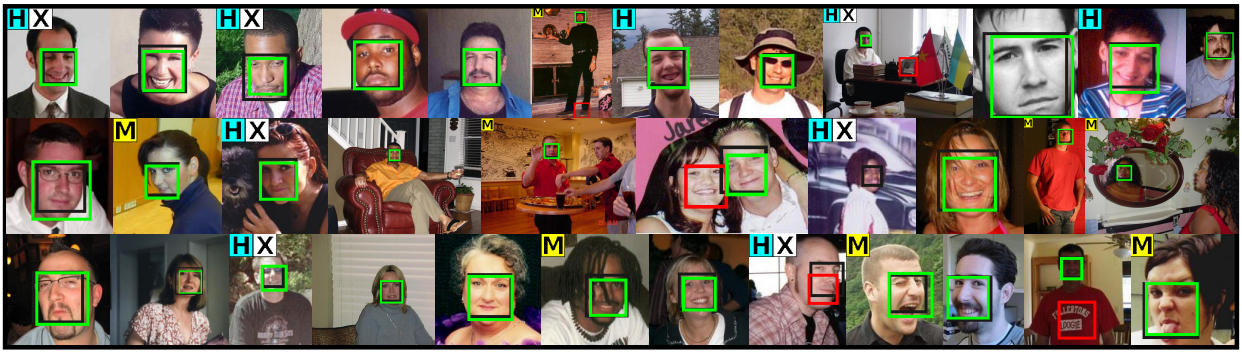

Fig. 13 Output of WiLFs on the GENKI dataset. Green rectangles are correct face detections, while red ones are false positives. The ground truth is indicated by black boxes. Letters H and M stand for human and machine annotations. Letter X denotes that the sample is an exemplar.

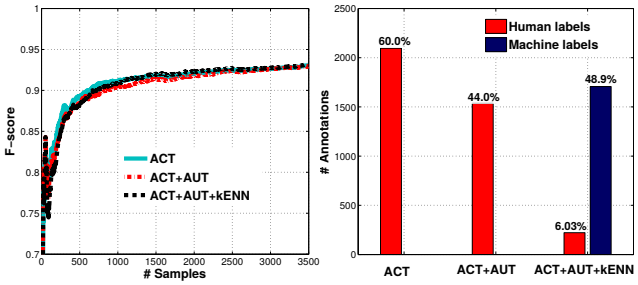

Fig. 14 Learning and detection performance of WiLFs on the GENKI face dataset according to their main components (ACT, AUT, and KENN). Left: Incremental detection performance. Right: Number of human and machine annotations.

Face Detection Rates on GENKI Dataset					
Detector	**Recall**	**Precision**	**F-score**	**Training**	**IoU**
SSH	98.7	93.7	96.2	Offline	0.4
SSH	89.6	85.1	87.3	Offline	0.5
WiLFs	90.2	95.6	92.8	Online	0.5

Table 6 Detection rates (%) of WiLFs and SSH detectors.

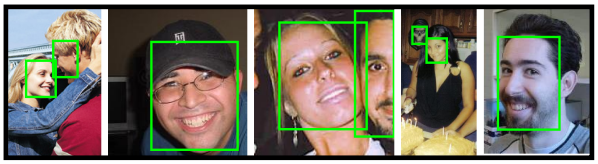

Fig. 15 Face detection results provided by [38].

Despite the excellent rates provided by deep learning, the works Li *et al.* [31] and Kumar *et al.* [29] obtain remarkable detection rates using a set of discriminative exemplar samples. These methods are related with the proposed approach but are computed offline, while WiLFs are continuously adding exemplar samples as new images are available.

Fig. 1 displays the response of the proposed method on the FDDB dataset. WiLFs are able to detect most faces in spite of hard conditions. However, the method has difficulty for detecting faces under occlusions and with large out-of-plane rotations. We assume that if multiple classifiers are trained by separated for each view, the detection performance would increase substantially. This idea was already addressed in [53] for multi-view object detection.

GENKI Dataset. The proposed method is also validated in the GENKI face dataset. WiLFs are computed and tested simultaneously across 3500 face images spanning a wide range of subjects, facial appearance, illumination, imaging conditions, and camera models [1]. Fig. 13 shows some example images. In this database, most images contain only one person and faces are mainly seen from a frontal view.

Similarly to Table 4, Fig. 14-left shows the learning and detection performance of the method in terms of its main components (ACT, AUT and κENN). We observe that all configurations present a similar behavior: the incremental detection rate (F-score) improves along with the number of samples used to compute WiLFs. Nevertheless, Fig. 14-right shows again that using κENN and the adaptive threshold (AUT), the amount of human intervention is drastically reduced. This becomes more evident in κENN where the percentage of human annotation drops from 44% (1540 images) to 6.03% (211 images).

To compare our approach in this dataset, we have tested a recent and state-of-the-art face detector based on deep learning. Particularly, we use the SSH detector[18] proposed by Najibi *et al.* [38] which provides outstanding detection rates on the FDDB dataset, refer to Fig. 12. This detector was tested over the 3500 dataset images using default settings.

Table 6 reports the face detection rates provided by SSH and WiLFs. Note that SSH is tested using two different Intersection over Union (IoU) thresholds during evaluation since the box predictions given by SSH differ substantially in size from the ground truth. Please look at Fig. 15 to see some examples. To avoid penalizing this detector, apart from the default threshold (0.5) we also considered a looser value (0.4) that results in very high detection rates. In particular, SSH obtains 98.7 of recall but a lower precision score (93.7). This is caused by a problem in the dataset annotation: only one face is annotated per image. Hence, other faces detected by SSH are considered as false positives, reducing thus the precision. In Fig. 15 we see that SSH detects multiple faces

[18] https://github.com/mahyarnajibi/SSH

Fig. 16 Detection results provided by WiLFs in the AFW face dataset.

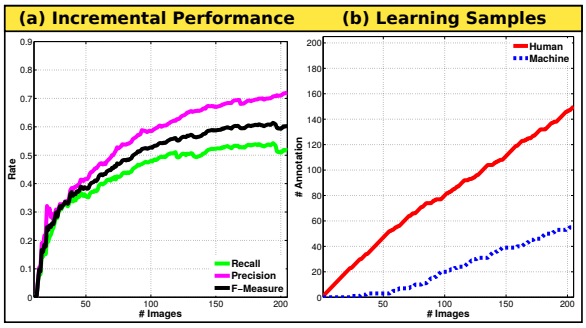

Fig. 17 Learning and detection performance of WiLFs on the AFW dataset.

and its robustness to out-of-plane rotations. This excellent performance is, however, at expense of a rigorous and offline training using a fully and manually annotated dataset, as well as the use of GPU cards.

Our method, on the other hand, achieves remarkable detection rates: 90.2 and 95.6 of recall and precision respectively. The gap in performance between the rates attained in FDDB and GENKI datasets lies in that latter is comprised mainly of frontal faces, showing again that WiLFs perform better for frontal views. In spite of that, the proposed method achieves good detection rates for a classifier that is trained incrementally (one sample at time) and efficiently using a CPU card, as well as requiring much less human supervision. Fig. 13 shows some examples with the response of WiLFs during the online face learning and detection.

AFW Dataset. WiLFs are also tested in the AFW face dataset [59]. It is a small dataset with 205 images containing faces under challenging conditions which is used for face detection, pose estimation, and landmark localization in the wild. Fig. 16 shows some examples of the output of WiLFs for face detection. Similar to the FDDB and GENKI datasets, the method is able to detect the most frontal faces but it exhibits certain difficulty for rotated faces. As suggested before this can be dealt using a multi-view approach.

The number of human annotations and the detection rates of WiLFs during training are visualized in Fig. 17. We see that the incremental detection values are relatively low because the database is small and the online classifier needs a substantial number of training samples to obtain good results. Note, for instance, that in Fig. 9-a WiLFs attain remarkable results after 500 image samples. The same occurs to the accumulative numbers of human and machine annotations, shown in Fig. 17-b, where the degree of human intervention decreases slightly since the dataset is small. However, we see that the number of machine annotations is growing steadily. Specifically, the final numbers of human and machine annotations are 147 samples (71.7%) and 58 samples (28.3%) respectively.

Since the AFW database is very small, the performance of WiLFs is quite inferior to other approaches in the state of the art. For comparison we have computed the Average Precision (AP) score which measure the face detection performance in terms of recall and precision for varying thresholds. Our method obtains an Average Precision (AP) score of 71.4 while other works attain very high scores. This is the case of Li *et al.* [32] and Mathias *et al.* [36] which obtain 96.7 and 97.1 respectively. However, these approaches were computed offline with larger datasets (e.g above 10k images). Contrary, we learn and test our classifier on the fly with AFW data only.

6 Conclusions

We have presented Wild Lady Ferns (WiLFs), an online and interactive detection approach that in contrast with other offline and time-consuming methods is capable of detecting faces using small human supervision. Despite the results obtained by WiLFs are distant from recent methods based on deep learning, the proposed approach allows to simultaneously learn and detect faces on the fly in about 1 frames per second without using temporal information or tracking ideas. In addition, WiLFs reduce gradually the amount of human intervention whereas conventional approaches and deep networks are mostly fully supervised, requiring of thousands or millions of human-made image annotations. As future work, we will study integrating further deep-learning techniques into WiLFs to increase the detection performance but keeping the efficiency and online computation of WiLFs.

References

1. Genki face dataset. `http://mplab.ucsd.edu/TheMPLabGENKIDatabase/GENKI-4KSubset`.
2. N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *International Conference on Machine Learning*, pages 1–9, 1998.

3. K. Ali and K. Saenko. Confidence-rated multiple instance boosting for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.
4. B. Babenko, M.H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 33(8):1619–1632, 2011.
5. Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2012.
6. Yu Cheng, Zhengzhang Chen, Lu Liu, Jiang Wang, Ankit Agrawal, and Alok Choudhary. Feedback-driven multiclass active learning for data streams. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1311–1320. ACM, 2013.
7. Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 7(2–3):81–227, 2012.
8. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, 2005.
9. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
10. P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.
11. Gonzalo Ferrer, Anaís Garrell, Michael Villamizar, Iván Huerta, and Alberto Sanfeliu. Robot interactive learning through human assistance. In *Multimodal Interaction in Image and Video Applications*, pages 185–203. Springer, 2013.
12. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.
13. J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, 2011.
14. M. Godec, P.M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. *Computer Vision and Image Understanding*, 117(10):1245–1256, 2013.
15. H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 260–267, 2006.
16. H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on–line boosting. In *British Machine Vision Conference*, 2006.
17. H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*, pages 234–247, 2008.
18. S. Hare, A. Saffari, and P.H. Torr. Efficient online structured output learning for keypoint-based object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 15, pages 1894–1901, 2012.
19. Peiyun Hu and Deva Ramanan. Finding tiny faces. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1522–1530. IEEE, 2017.
20. Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.
21. Vidit Jain and Erik Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, University of Massachusetts, Amherst, 2010.
22. Vidit Jain and Erik Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 577–584. IEEE, 2011.
23. Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–56, 2010.
24. T.K. Kim and R. Cipolla. Mcboost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In *Neural Information Processing Systems*, pages 841–848, 2009.
25. T.K. Kim, T. Woodley, B. Stenger, and R. Cipolla. Online multiple classifier boosting for object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, volume 11, pages 1–6, 2010.
26. Martin Köstinger, Paul Wohlhart, Peter M Roth, and Horst Bischof. Robust face detection by simple means. In *DAGM 2012 CVAW workshop*, 2012.
27. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
28. E. Krupka, A. Vinnikov, B. Klein, A.B. Hillel, D. Freedman, and S. Stachniak. Discriminative ferns ensemble for hand pose recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3670–3677, 2014.
29. Vijay Kumar, Anoop Namboodiri, and CV Jawahar. Visual phrases for exemplar face detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1994–2002, 2015.
30. D. Lewis and W. Gale. A sequential algorithm for training text classifiers. *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.
31. Haoxiang Li, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Gang Hua. Efficient boosted exemplar-based face detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1843–1850, 2014.
32. Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015.
33. Yuxuan Li and Xiuzhen Zhang. Improving k nearest neighbor with exemplar generalization for imbalanced classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 321–332. Springer, 2011.
34. Baozhen Liu, Hang Wu, Weihua Su, Wenchang Zhang, and Jinggong Sun. Rotation-invariant object detection using sector-ring hog and boosted random ferns. *The Visual Computer*, 34(5):707–719, 2018.
35. T. Malisiewicz, A. Gupta, and A.A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *International Conference on Computer Vision*, pages 89–96, 2011.
36. Markus Mathias, Rodrigo Benenson, Marco Pedersoli, and Luc Van Gool. Face detection without bells and whistles. In *European conference on computer vision*, pages 720–735. Springer, 2014.
37. K. P. Murphy. *Machine learning: a probabilistic perspective*. 2012.
38. Mahyar Najibi, Pouya Samangouei, Rama Chellappa, and Larry Davis. SSH: Single stage headless face detector. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017.
39. M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 32(3):448–461, 2010.
40. Je-Kang Park and Dong-Joong Kang. Unified convolutional neural network for direct facial keypoints detection. *The Visual Computer*, pages 1–12, 2018.

41. Wei Quan, Jim X Chen, and Nanyang Yu. Robust object tracking using enhanced random ferns. *The Visual Computer*, 30(4):351–358, 2014.
42. Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. IEEE, 2011.
43. Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1):157–173, 2008.
44. J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. Prost: Parallel robust online simple tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 723–730, 2010.
45. Santi Segui, Michal Drozdzal, Petia Radeva, and Jordi Vitria. An integrated approach to contextual face detection. In *ICPRAM*, 2012.
46. B. Settles. Active learning literature survey. *University of Wisconsin, Madison 52(55-56) 2010: 11*, 2010.
47. H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory*, pages 287–294, 1992.
48. P. Sharma and R. Nevatia. Multi class boosted random ferns for adapting a generic object detector to a specific video. In *IEEE Winter Conference on Applications of Computer Vision*, pages 745–752, 2014.
49. A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 29(5):854–869, 2007.
50. M. Villamizar, J. Andrade-Cetto, A. Sanfeliu, and F. Moreno-Noguer. Bootstrapping boosted random ferns for discriminative and efficient object classification. *Pattern Recognition*, 45(9):3141–3153, 2012.
51. M. Villamizar, J. Andrade-Cetto, A. Sanfeliu, and F. Moreno-Noguer. Boosted random ferns for object detection. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 2017.
52. M. Villamizar, A. Garrell, A. Sanfeliu, and F. Moreno-Noguer. Online human-assisted learning using random ferns. In *International Conference on Pattern Recognition*, pages 2821–2824, 2012.
53. M. Villamizar, H. Grabner, J. Andrade-Cetto, A. Sanfeliu, L. Van Gool, and F. Moreno-Noguer. Efficient 3d object detection using multiple pose-specific classifiers. In *British Machine Vision Conference*, 2011.
54. Michael Villamizar, Anaís Garrell, Alberto Sanfeliu, and Francesc Moreno-Noguer. Interactive multiple object learning with scanty human supervision. *Computer vision and image understanding*, 149:51–64, 2016.
55. Michael Villamizar, Alberto Sanfeliu, and Francesc Moreno-Noguer. Fast online learning and detection of natural landmarks for autonomous aerial robots. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4996–5003. IEEE, 2014.
56. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages l–511, 2001.
57. A. Yao, J. Gall, C. Leistner, and L. van Gool. Interactive object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3242–3249, 2012.
58. B. Zeisl, C. Leistner, A. Saffari, and H. Bischof. On-line semi-supervised multiple-instance boosting. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1879–1879, 2010.
59. Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012.