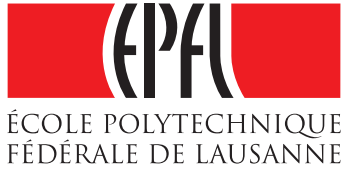


KernelBoost: Supervised Learning of Image Features For Classification*

Carlos Becker (carlos.becker@epfl.ch)
http://cvlab.epfl.ch/~becker

Roberto Rigamonti (roberto.rigamonti@epfl.ch)
http://cvlab.epfl.ch/~rigamont

Vincent Lepetit (vincent.lepetit@epfl.ch)
http://cvlab.epfl.ch/~lepetit

Pascal Fua (pascal.fua@epfl.ch)
http://cvlab.epfl.ch/~fua

School of Computer and Communication Sciences
Swiss Federal Institute of Technology, Lausanne (EPFL)

Technical Report

February 04, 2013

* This work has been supported in part by the Swiss National Science Foundation.

Abstract. We propose a fully-supervised approach to training classifiers that automatically learn features directly from image data. This drops the dependency on hand-designed filters and features, which is generally a trial-and-error process and often yields far-from-optimal results. Our approach relies on the Gradient Boosting framework, learning discriminative features at each stage in the form of convolutional filters. It depends on just few easy-to-tune parameters, it is simple and general, and we show it outperforms state-of-the-art methods in tasks ranging from pixel classification in very different types of images to object detection.

1 Introduction

Many of the best Computer Vision algorithms proposed to solve problems ranging from image classification to object detection rely on two key ingredients: a set of trained or hand-designed features, and a classifier trained on these features. While this combination has produced strong results in many different cases, there is no guarantee that the features perfectly meet the requirements of the classifier that uses them. This has usually been addressed by parametrizing the feature extractors and allowing the classifier to choose the best parameters. However, this can only work if the features have been chosen appropriately, so suitable feature types must be sought, often by trial-and-error.

In this paper, we propose an approach to learning the features and building the classifier simultaneously, thus guaranteeing that the best possible features are used at every stage of the computation. We will show that this leads to increased performance with respect to state-of-the-art approaches for a broad range of applications, while at the same time relieving the human designer from the need to select specific features and their parametrization.

To this end, we introduce a novel, fully-discriminative method that fits naturally into the Gradient Boosting framework [9,32], but could also be incorporated, for instance, in Random Forests [3]. Our main contribution is an original approach to learning both the weak learners and the boosting weights at the same time, so that the weak classifiers we use at every boosting stage truly are as good as they can be.

More specifically, our weak classifiers rely on convolutional filters whose kernels are learned during boosting. Arguably, this could be described as learning the classifier parameters as it is often done by boosting algorithms. However, because the parameter space is so large, a standard boosting approach, such as a grid search, would be impractical. Instead, we compute the kernels in closed form, which allows us to handle the enormous size of the parameter space. Convolutional Neural Networks [15] and Deep Belief Networks (DBNs) [10,16,22] also simultaneously learn convolutional features and classification weights. DBNs in particular do so in an unsupervised way and only use discriminative information for fine-tuning. This may not be ideal since it has recently been shown that fully-supervised versions of these approaches have a large unexploited potential

and can significantly outperform competing methods in several difficult cases [5]. However, the neural network architecture adopted in [5] requires specialized set-up and careful design. Furthermore, it is computationally expensive to train, even on Parallel GPUs. By contrast, our approach is much less demanding, can be run on ordinary desktop machines, and involves just few, easily-tunable parameters.

Our contribution is therefore a new, fully-supervised, practical approach to jointly learning the filters our weak learners rely on and the weights and parameters of the classifier that uses them. As a result, optimized features are used at each stage of the iterative learning procedure. We will show that classifiers trained in this manner outperform the state-of-the-art on medical image segmentation, boundary detection, and car detection, thus demonstrating the versatility of our approach.

2 Related Work

Most boosting-based approaches build a classifier by iteratively selecting features from a predefined set. Common features include Haar wavelets [28] and Local Binary Patterns [2], which are very efficient to compute. This efficiency is important at training time to be able to explore a large amount of such features at every boosting iteration. Features can also be image gradients, integrated over rectangular regions [17] or predefined bins [6,33], which have shown to be more discriminative for tasks such as object detection.

Selecting features also involves sampling the space of parameters required to compute them, either randomly or at regular intervals. This is only possible because these features are designed to have relatively few such parameters, which also limits their discriminative power. For example, they are typically computed over rectangular blocks, possibly rotated [18].

More general features in the form of linear projections applied to the input image are introduced in [29]. The space of projections is too large to be sampled, but they can be instead found in closed-form using discriminant analysis techniques [9]. Unfortunately, this approach is both slow and prone to overfitting, a common issue in discriminant analysis. This is accentuated in tasks such as object detection, where objects span over large areas, increasing projection dimensionality. In our approach we also compute our features in closed form, albeit using a different technique we will show to be more effective. A primary difference is that we combine random sampling and closed-form estimation of convolutional linear filters over small parts of the images, thereby avoiding overfitting and reducing computational complexity. Moreover, we introduce a regularization to guarantee smooth filters. As shown in the results section, this improves generalization.

Convolutional Neural Networks (CNNs) represent another approach to learning features in the form of linear convolutional filters. CNNs are multi-layer networks trained in a supervised way [15]. Some layers are hardcoded to perform pooling operations while the others are automatically optimized and can be interpreted as linear convolutional filters. Even though CNNs can yield very strong

results, the optimization problem is enormous and the optimizer is likely to get trapped in a local minima. In some cases, this can be alleviated by hardcoding the first layer to extract image gradients [20], but this is not a general solution.

The difficulty of optimizing deep networks in a supervised and discriminative way has spurred the emergence of unsupervised and generative approaches to feature learning. For example, Restricted Boltzmann Machines [10], which can be used to extract features from images, are trained by seeking to approximate the distribution of natural images. More recent approaches [19,12,31,22], motivated by the work of Olshausen and Field [21], seek a representation of the input images that is constrained to be sparse, and from which the input images can be later reconstructed. Such approaches can then be used to efficiently build deep architectures by training each layer, one after the other, to reconstruct the output of the previous layer [10].

However, it was recently shown that Deep Neural Networks (DNNs) could in fact be trained in a supervised way by using GPUs [5], even when employing broader layers than traditional Convolutional Networks. Because the algorithm still has to contend with the many local minima of the objective function, several DNNs are trained independently and averaged to mitigate this problem. Our own approach is also supervised, but we rely on a much simpler architecture that can be trained on an ordinary computer. Moreover, our method relies on just few easily-tunable parameters, as demonstrated in the results section.

3 Gradient Boosting

Since our approach relies on the Gradient Boosting framework, we outline here its main characteristics. Our contribution is presented in the next section.

Gradient Boosting is a an approach to approximate a function $\varphi^* : \mathbb{R}^n \to \mathbb{R}$ by a function φ of the form

$$\varphi(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j \, h_j(\mathbf{x}) \,, \tag{1}$$

where the $\alpha_j \in \mathbb{R}$ are real-valued weights, $h_j : \mathbb{R}^n \to \mathbb{R}$ are weak learners, and $\mathbf{x} \in \mathbb{R}^n$ is the input vector.

Gradient Boosting can be seen as a generalization of AdaBoost. The latter is designed to minimize the exponential loss with stump weak learners $h_j : \mathbb{R}^n \to \{+1, -1\}$, while Gradient Boosting can make use of real-valued weak learners and is able to minimize other loss functions [9]. Gradient Boosting has shown significant performance improvements in many classification problems with respect to classic AdaBoost [4].

Given training samples $\{(\mathbf{x}_i, y_i)\}_{i=1,..,N}$, where $\mathbf{x}_i \in \mathbb{R}^n$ and $y_i = \varphi^*(\mathbf{x}_i)$, $\varphi(\cdot)$ is constructed in a greedy manner by iteratively selecting weak learners and their weights to minimize a loss function $\mathcal{L} = \sum_{i=1}^{N} L(y_i, \varphi(\mathbf{x}_i))$. In our work we use a formulation based on a quadratic approximation of the gradient descent direction in function space [32]. The corresponding algorithm is given in Alg. 1 and can be employed to minimize any twice-differentiable loss function. At

Algorithm 1 Gradient Boosting with Quadratic Approximation [32]

Input: Training samples and labels $\{(\mathbf{x}_i, y_i)\}_{i=1,..,N}$
Number of iterations M
Shrinkage factor $0 < \gamma \leq 1$

1: Set $\varphi_0(\cdot) = 0$

2: **for** $j = 1$ to M **do**

3: Let $w_i^j = \frac{\partial^2 L(y_i,\phi)}{\partial \phi^2}\Big|_{\phi=\varphi_{j-1}(\mathbf{x}_i)}$ and
$r_i^j = -\frac{1}{w_i^j}\frac{\partial L(y_i,\phi)}{\partial \phi}\Big|_{\phi=\varphi_{j-1}(\mathbf{x}_i)}$

4: Find weak learner
$h_j(\cdot) = \underset{h(\cdot)}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^j \left(h(\mathbf{x}_i) - r_i^j\right)^2$

5: Find α_j through line search
$\alpha_j = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{N} L\big(y_i, \varphi_{j-1}(\mathbf{x}_i) + \alpha\, h_j(\mathbf{x}_i)\big)$

6: Let $\varphi_j(\cdot) = \varphi_{j-1}(\cdot) + \gamma\, \alpha_j\, h_j(\cdot)$

7: **end for**

8: **return** $\varphi_M(\cdot)$

each iteration j, Gradient Boost seeks for a weak learner to regress the gradient descent direction r_i^j at each sample $\mathbf{x}_i$, minimizing the loss function $\mathcal{L}$ over the training data. Commonly used classification loss functions are the exponential loss $L(y_i, \varphi(\mathbf{x}_i)) = e^{-y_i \varphi(\mathbf{x}_i)}$ and the log loss $L(y_i, \varphi(\mathbf{x}_i)) = \log(1 + e^{-2 y_i \varphi(\mathbf{x}_i)})$.

The weak learners $h_j(\cdot)$ are generally either decision stumps [28] or regression trees [9]. Regression trees are a generalization of decision stumps and usually yield significantly better performance [9], achieving state-of-the-art in many classification problems [4]. Regression trees are typically learned in a greedy manner, building them one split at a time starting from the root [9].

4 Proposed Approach

Assume that we are given training samples $\{(\mathbf{x}_i, y_i)\}_{i=1...N}$, where $\mathbf{x}_i \in \mathbb{R}^n$ represents an image or a patch and $y_i \in \{-1, 1\}$ its label. Our goal is to simultaneously learn both the features and a function $\varphi(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ based on these features to predict the value of y corresponding to previously unseen $\mathbf{x}$.

We first recall below how decision stumps and regression trees can be built to optimize a Gradient Boosting classifier. We then describe how we learn relevant features while growing the trees.

4.1 Growing Regression Trees

The standard approach to implementing the Gradient Boosting procedure summarized by the pseudo-code of Alg. 1 searches through sets of weak learners that rely on a fixed set of features, such as Haar wavelets [28]. At each iteration j, it selects the $h_j(\cdot)$ that minimizes

$$\sum_{i=1}^{N} w_i^j \left(h(\mathbf{x}_i) - r_i^j\right)^2 \tag{2}$$

which appears as Step 4 of Alg. 1. In our approach, we simultaneously learn the image features as well, instead of selecting them from a predefined set.

We consider here weak learners that are regression trees based on convolutions of the image or image patch $\mathbf{x}$ with a set of learned convolution kernels $\mathcal{K}_j$. We write them as $h_j(\mathbf{x}) = T(\theta_j, \mathcal{K}_j, \mathbf{x})$ where θ_j represents the tree parameters. In essence, standard approaches learn only the θ_j, while we also learn the kernels $\mathcal{K}_j$. An example regression tree is shown in Fig. 1.

The tree learning procedure is performed one split at a time, as in [9]. A split consists of a test function $t(\cdot) \in \mathbb{R}$, a threshold τ, and return values η_1 and η_2. Its prediction function can be written as

$$s(\cdot) = \begin{cases} \eta_1 \text{ if } t(\cdot) < \tau \\ \eta_2 \text{ otherwise.} \end{cases} \tag{3}$$

Given a test function $t(\cdot)$, the optimal root split at iteration j is found by minimizing

$$\sum_{i|t(\mathbf{x}_i)<\tau} w_i^j \left(r_i^j - \eta_1\right)^2 + \sum_{i|t(\mathbf{x}_i)\geq\tau} w_i^j \left(r_i^j - \eta_2\right)^2 , \tag{4}$$

where τ, η_1, and η_2 are typically found through exhaustive search [9].

In our approach, we introduce a test function that operates on the results of $\mathbf{x}_i$ and a kernel $\mathbf{k}$, namely $t(\mathbf{x}_i) = \mathbf{k}^\top \mathbf{x}_i$. Therefore, learning a split in our framework involves searching for a kernel $\mathbf{k}$, leaf values η_1 and η_2, and split point τ that minimize

$$\sum_{i|\mathbf{k}^\top\mathbf{x}_i<\tau} w_i^j \left(r_i^j - \eta_1\right)^2 + \sum_{i|\mathbf{k}^\top\mathbf{x}_i\geq\tau} w_i^j \left(r_i^j - \eta_2\right)^2 . \tag{5}$$

Since the space of all possible kernels is enormous, we perform this minimization in stages. Our approach is described in Alg. 2: we first construct a set of kernel candidates, then for each candidate we find the optimal τ through exhaustive search. For a given pair of kernel $\mathbf{k}$ and threshold τ, the optimal values for η_1 and η_2 are then simply found as the weighted average of the r_i^j values of the $\mathbf{x}_i$ samples that fall on the corresponding side of the split.

This parameter selection step for η_1, η_2, and τ is standard [9] but the kernel learning is not, as we consider a much more general form for the kernels $\mathbf{k}$ than is usually done. We now describe this step in detail.

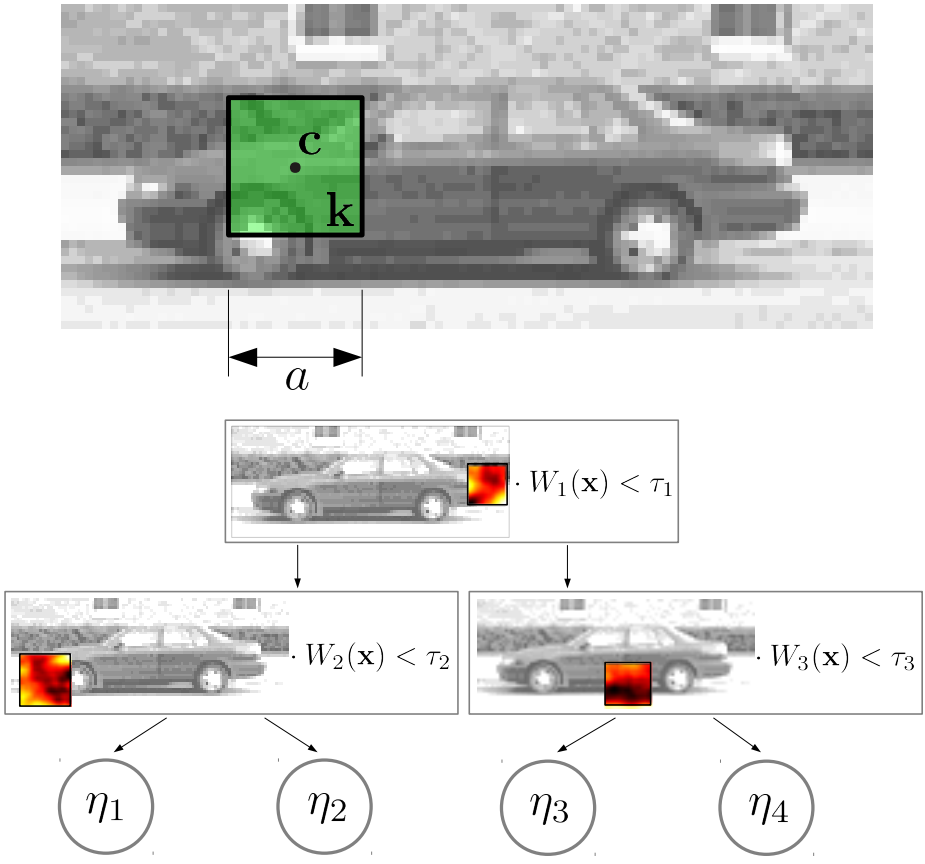


Fig. 1. (top) Individual kernel localization example and **(bottom)** example two-level tree with learned kernels from the car detection dataset [1], where $\eta_1, \eta_2, \eta_3, \eta_4$ are leaf values and W_i represents a sub-window extracted from $\mathbf{x}$.

4.2 Learning Convolution Kernels

To make computations tractable, we restrict the kernels $\mathbf{k}$ to being square windows within $\mathbf{x}$. This remains more general than most previous methods while reducing the dimensionality of the problem and allowing our splits to focus on local image features.

Let us introduce an operator $W_{\mathbf{c},a}(\mathbf{x})$ that returns, in vector form, the pixel values of $\mathbf{x}$ within a square window centered at $\mathbf{c}$ with side length a, as in Fig. 1. The criterion of Eq. (2) becomes

$$\sum_{i=1}^{N} w_i^j \left(\mathbf{k}^\top W_{\mathbf{c},a}(\mathbf{x}_i) - r_i^j\right)^2 , \tag{6}$$

where the kernel $\mathbf{k}$ is now restricted to a square window parametrized by $\mathbf{c}$ and a. For a given pair of $\mathbf{c}$ and a, we can compute the optimal $\mathbf{k}$ in closed form

by solving the least-squares problem of Eq. (6). However, applying this method directly tends to overfit, as seen in our experiments. We therefore introduce two refinements:

1. **Regularization.** We favor smooth kernels by introducing a regularization term into the criterion of Eq. (6), which becomes:

$$\sum_i w_i^j \left(\mathbf{k}^\top W_{\mathbf{c},a}(\mathbf{x}_i) - r_i^j\right)^2 + \lambda \sum_{(m,n)\in\mathcal{N}} \left(\mathbf{k}^{(m)} - \mathbf{k}^{(n)}\right)^2 , \tag{7}$$

   where $(m,n) \in \mathcal{N}$ are pairs of indices that correspond to neighboring pixels and $\mathbf{k}^{(m)}$ is the m^{th} pixel of kernel $\mathbf{k}$. The second term in Eq. (7) imposes a smooth kernel, controlled by $\lambda > 0$. Note that Eq. (7) can be minimized in closed form using least squares.
2. **Splitting the training set.** We randomly split the training set in two equal-sized subsets T_1 and T_2. We use T_1 to minimize the criterion of Eq. (7) and find kernel $\mathbf{k}$, while T_2 is employed to learn the τ, η_1, and η_2 parameters that minimize the criterion of Eq. (5).

As described in Alg. 2, we repeat this operation for many randomly selected values of $W_{\mathbf{c},a}(\mathbf{x})$, λ, T_1, and T_2 to select the split that returns the smallest value for the criterion of Eq. (5). The recursive splitting procedure of Alg. 2 then produces trees that are used as weak learners in Alg. 1.

Note that, when using the exponential loss, we have $r_i^j \in \{-1, 1\}$ (line 3 of Alg. 1). In this case and when no regularization is imposed, that is, $\lambda = 0$, the $\mathbf{k}$ that minimizes Eq. 6 would be identical to the one returned by LDA [9], up to a scale factor. However, this is a particular case of our formulation, which instead allows for smoothing as well as more outlier-robust losses such as the log loss. Smoothing yields higher generalization, while outlier-robust losses are essential to deal with tasks such as pixel classification.

4.3 Learning more Complex Features

For some of the problems we consider, we also introduce a technique related to *Auto-Context* [27]. Auto-Context is a simple and effective way to increase classification performance, training a chain of Q sequential classifiers, such that classifier $\mathcal{C}_q$ relies on features computed from both the original training image and the predictions of classifier $\mathcal{C}_{q-1}$ [27]. This is easily implemented in our framework by treating each image or image patch as two channels: the image data itself, and the prediction image from classifier $\mathcal{C}_{q-1}$. The only modification to the algorithm of Section 4.2 is that the split learning algorithm Alg. 2 now searches kernels for both channels.

Thus, unlike in the original Auto-Context approach of [27] that is limited to a discrete subset of features, our algorithm can learn arbitrarily complex ones. We show in the results section that this can enhance performance.

Algorithm 2 KernelBoost Split Learning

Input: Training samples $\{\mathbf{x}_i\}_{i=1,..,N}$
Weights and responses $\{w_i, r_i\}_{i=1,..,N}$ at boosting iteration j, as in Alg. 1
Number P of kernels to explore
Set $\mathbb{W}$ of possible window locations and sizes
Set $\mathbb{L}$ of possible regularization factors

Randomly divide training set in two sets T_1 and T_2

// Phase I: kernel search on T_1

for $p = 1$ to P **do**
  Pick random window $W_{\mathbf{c}_p,a_p} \in \mathbb{W}$
  Pick random regularization factor $\lambda_p \in \mathbb{L}$
  Find kernel $\mathbf{k}_p$:

$$\mathbf{k}_p = \underset{\mathbf{k}}{\operatorname{argmin}} \sum_{i \in T_1} w_i \left(\mathbf{k}^\top W_{\mathbf{c}_p,a_p}(\mathbf{x}_i) - r_i \right)^2 + \lambda_p \sum_{(m,n) \in \mathcal{N}} \left(\mathbf{k}^{(m)} - \mathbf{k}^{(n)} \right)^2$$

end for

// Phase II: split search on T_2

for $p = 1$ to P **do**
  Let $W_p(\cdot) = W_{\mathbf{c}_p,a_p}(\cdot)$
  Find $\tau_p, \eta_{1,p}, \eta_{2,p}$ for $\mathbf{k}_p$ through exhaustive search on T_2:

$$\tau_p, \eta_{1,p}, \eta_{2,p} = \underset{\tau,\eta_1,\eta_2}{\operatorname{argmin}} \sum_{i|\mathbf{k}_p^\top W_p(\mathbf{x}_i) < \tau} w_i \left(r_i - \eta_1 \right)^2 + \sum_{i|\mathbf{k}_p^\top W_p(\mathbf{x}_i) \geq \tau} w_i \left(r_i - \eta_2 \right)^2$$

  Compute split cost on T_2:

$$\epsilon_p = \sum_{i|\mathbf{k}_p^\top W_p(\mathbf{x}_i) < \tau_p} w_i \left(r_i - \eta_{1,p} \right)^2 + \sum_{i|\mathbf{k}_p^\top W_p(\mathbf{x}_i) \geq \tau_p} w_i \left(r_i - \eta_{2,p} \right)^2$$

end for

return $(\mathbf{k}_p, \tau_p, \eta_{1,p}, \eta_{2,p})$ that yields the smallest ϵ_p.

4.4 Training Time

To make training faster, we modify the above algorithm as follows:

– At boosting iteration j, instead of searching for kernels at every tree split with Alg. 2, we run Phase I of Alg. 2 only once to generate a set of candidate kernels at the tree root. These kernels are then used to train as many splits as needed to construct the tree.

– The resulting least-squares problem in line 5 of Alg. 2 may grow very large in tasks such as pixel classification. Therefore, we reduce T_1 by uniformly sampling N_R samples from it. In our experiments we set $N_R = 10000$.

In practice, these modifications yield a 10x speed up during training, with a negligible impact on accuracy. For the pixel classification datasets presented in the results section, training took 7 hours for $M = 2000$ boosting iterations on a 12-core computer.

4.5 Parameters

In all the experiments reported in Section 5, tree depth is limited to 3 levels and shrinkage is set to $\gamma = 0.1$. These are standard values used in Gradient Boosting [9].

For the pixel classification tasks we use the log loss to increase robustness to outliers [9]. For the object detection task, we rely on the exponential loss as in [28].

The only two parameters that were varied between datasets are (a) the patch size, which was chosen according to the scale of the structures or objects of interest, and (b), the set of regularization values $\mathbb{L}$. We visually choose the minimum regularization value $\lambda_{\min}$ as the value that yields filters at the first boosting iteration as smooth as the images $\mathbf{x}$. The rest of the regularization values are set to multiples of $\lambda_{\min}$.

The number of boosting iterations M was set to 2000 when not using Auto-Context. When using Auto-Context, we set $M = 500$ for each classifier $\mathcal{C}_q$, yielding a total of $500 \cdot Q$ trained trees. In all cases we explore $P = 100$ kernels per iteration and maximum kernel size is 19×19.

The reduced number of parameters to define is a key advantage of our approach. We show that we are able to outperform state-of-the-art methods without the need for parameter tuning.

5 Results

We evaluated our approach both on pixel classification in many different types of images and on object detection. As discussed in Section 4.5, we used the same parameters in all cases except for patch size, $\lambda_{\min}$ and loss function.

5.1 Pixel Classification

We use very different classes of images—retinal scans, confocal microscopy images, and electronic microscopy images—that all contain tubular structures and for which we have ground truth. To label each pixel of an input image as either being part of a linear structure or not, we take the sample vectors $\mathbf{x}$ to be image patches centered on individual pixels. Once our classifier is trained, it is possible to obtain a probability estimate for the label of a pixel centered at $\mathbf{x}$ as $p(y = 1|\mathbf{x}) = (1 + e^{-2\varphi(\mathbf{x})})^{-1}$ [9].

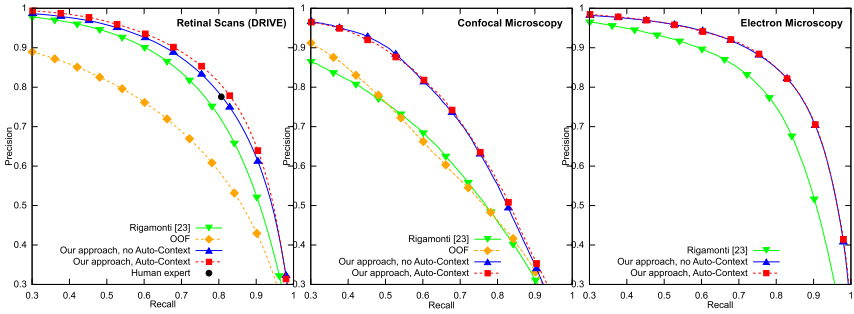


Fig. 2. Precision-recall curves for pixel classification. Our approach significantly outperforms all baselines, without the need for hand-designed features or parameter tuning. Auto-Context boosts performance in the DRIVE dataset, while obtaining similar results to no Auto-Context in the other two datasets.

We compare the results of our approach against those obtained by the Optimally Oriented Flux (OOF) filter [14] and those of [23]. The former is a handcrafted filter, widely acknowledged as being very good for delineating tubular structures. The latter is a new, hybrid approach that complements hand-crafted features with features learned in an unsupervised fashion, and achieves state-of-the-art performance. In our experiments with this kind of mixed features, Random Forests [3] delivered better results than Boosted Trees, which is consistent with what is reported in literature. For the electron microscopy case, where no tubular structures are present in the images and therefore we could not rely on any handcrafted method suited for the task, we used a pure learning-based approach. To this end, we learned a filter bank composed of 49 convolutional filters using unsupervised ℓ_1-based minimization, as in [23]. The parameters of the algorithms we compare against were all tuned to achieve their best performance, in order to provide a fair comparison.

We now describe each dataset in detail.

Retinal Scans The DRIVE dataset [25] is a publicly-available set of 40 RGB retinal scans where the aim is to segment blood vessels, for automated diagnosis purposes. Since we have two different ground truth sets from two different ophthalmologists, it is possible to estimate the score a human expert would achieve in the segmentation task.

Fig. 2 shows that approach outperforms the baselines, achieving human performance when not using Auto-Context. With Auto-Context, our approach becomes the only method to outperform human performance in this dataset up to the date of this publication.

Confocal Microscopy Images The brightfield dataset is made of four 2D minimum-intensity projections of bright-field micrographs of dyed neurons. Fig. 5 shows an example of these images, along with ground truth. The images are very complex in that both the staining process and the projections introduce significant structured and unstructured noise components. Also, the images are quite small compared with, for example, those of [23], although the ground truth is very accurate

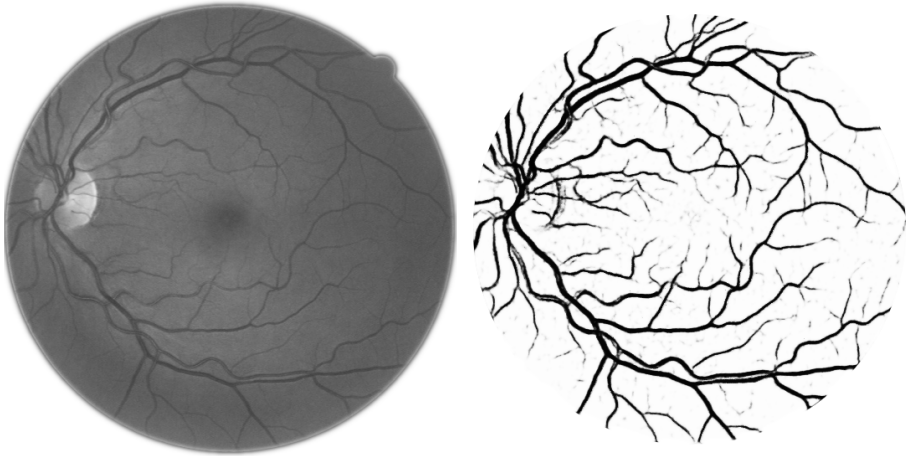


Fig. 3. Retinal Scans: **(left)** example image and **(right)** probability map $p(y = 1|x)$ obtained with our approach.

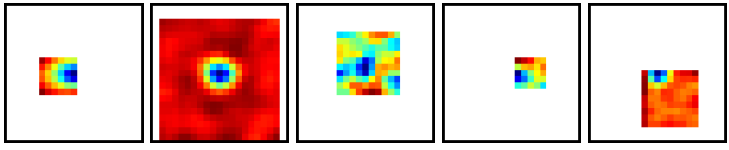


Fig. 4. Example filters found on the DRIVE blood vessel segmentation dataset (best viewed in color). The black box represents the patches $\mathbf{x}_i$. The filters only operate on the colored pixels.

in our case. These factors make this task particularly difficult for learning-based approaches, both ours and that of [23], as the scarcity of training samples made the algorithms prone to overfitting. We used two images for training, and the other two for testing.

Our approach outperforms the baselines, as shown in Fig. 2. Note that Auto-Context is neither beneficial nor detrimental in this case, which may be due to the scarcity of training data. On the other hand, this suggests that our approach is resistant to overfitting, a known characteristic of boosting-based techniques [9].

Electron Microscopy Images We also evaluated the capabilities of our method on the detection of grain boundaries in Electron Microscope images [7], an important task in material science. Fig. 6 shows one of the four high-resolution images of the dataset we created of thin layers of Zinc Oxide (ZnO), used in thin-film solar cells. The methods currently used in the industry to automatically obtain such detections rely either on simple gray-level thresholding or on watershed binarization [7], and are therefore not suitable for dealing with such images that constitute a severe challenge even for humans.

Ground truth delineation was made by a field expert. Since we are mostly interested in the detection of the boundaries and not in their exact delineation,

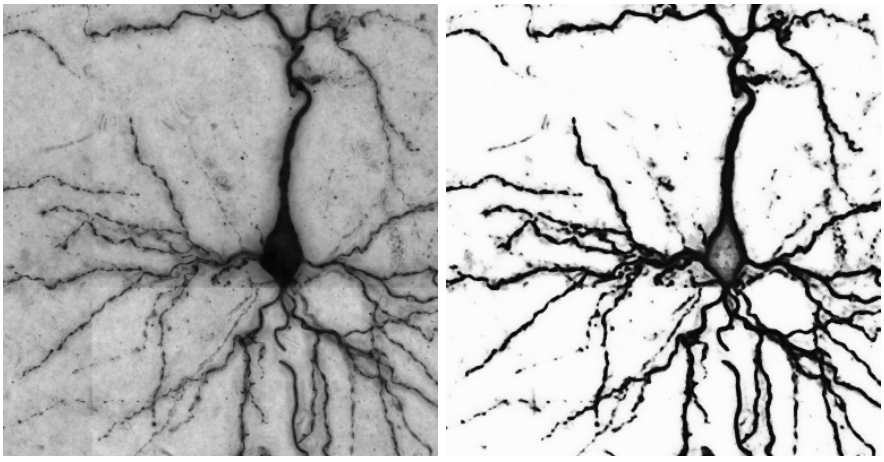


Fig. 5. Confocal Microscopy: **(left)** example image and **(right)** probability map $p(y = 1|x)$ obtained with our approach.

the pixels in a three-pixels wide region around the boundaries present in the ground-truth are ignored in the evaluation.

Our approach outperforms the baselines in this dataset as well, as shown in Fig. 2. Similar conclusions as with the Confocal Microscopy dataset can be drawn regarding Auto-Context.

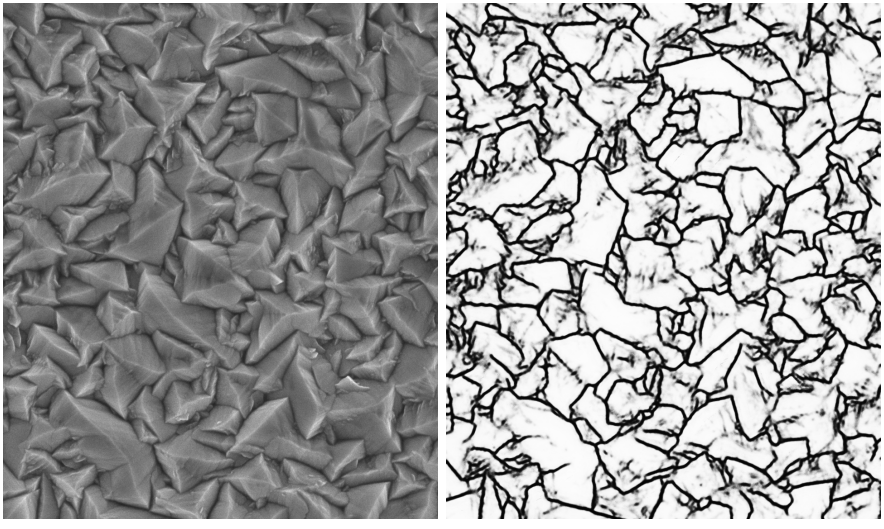


Fig. 6. Electron Microscopy: **(left)** example image and **(right)** probability map $p(y = 1|x)$ obtained with our approach.

Table 1. Performance on the UIUC Car Detection dataset at recall-precision equal-error-rate (EER) in the single-scale test set using $\mathbf{x}$ of the form $\mathbf{x}_{\mathrm{A}}$ with different values of smooth penalty λ.

Filter-search method (line 5 in Alg. 2)	Recall at EER
(a) LDA (no spatial regularization)	98.5%
(b) Least-squares, $\lambda = 0$	98.5%
(c) Least-squares, $\lambda = 250$	99.0%
(d) Least-squares, $\lambda = 1200$	100%
(e) Least-squares, $\lambda \in \mathbb{L}$ with $\lambda_{\min} = 1200$	100%

5.2 Car detection

We evaluated our algorithm on the UIUC car detection dataset [1]. The dataset is composed of 550 car and 500 non-car grayscale training image patches and the task is to learn to detect cars from side-views. Performance is evaluated on two separate image sets. The first is a single-scale test set, made out of 170 images containing 200 cars at approximately the same scale, while the second one is a multi-scale test set with 108 images and 139 cars at different scales.

The evaluation process is standardized and code has been provided by the authors to allow for comparison between published results.

The sample vectors $\mathbf{x}$ are made of the image gradient magnitudes for several quantized directions, since they are known to be a strong cue for object recognition. More formally, we tried $\mathbf{x}$ of the form:

- $\mathbf{x}_{\mathrm{A}}$: gradient magnitudes for orientations in $\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$, over all pixel locations.
- $\mathbf{x}_{\mathrm{B}}$: gradient magnitudes for orientations in $\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$ and the intensities from the image itself, over all pixel locations, using mean-variance normalization to mitigate illumination and contrast variations.

Table 1 presents the performance obtained on the single-scale test set with and without spatial regularization in the least-squares problem of Eq. (7). In all cases but (e), we fix λ to assess the influence of increasing regularization values. Doing so increases performance as well, obtaining perfect classification with $\lambda = 1200$ or when employing our full approach in (e), that picks $\lambda \in \mathbb{L}$ at random instead. As discussed in Sec. 4.2, we could also compute the kernels of Eq. (7) by LDA and should obtain exactly the same filters and results as those without regularization, which is indeed what happened when we performed the actual experiment. This shows the first advantage of our approach with respect to [29] which relies on discriminant analysis but lacks the ability to impose smooth filters. Moreover, [29] finds filters on the whole patch $\mathbf{x}$, which proved computationally intractable for this dataset, while our approach works on smaller sub-windows, making training feasible.

Table 2 presents the performance on the single and multi-scale test sets, compared against other algorithms found in the literature. Our method was trained

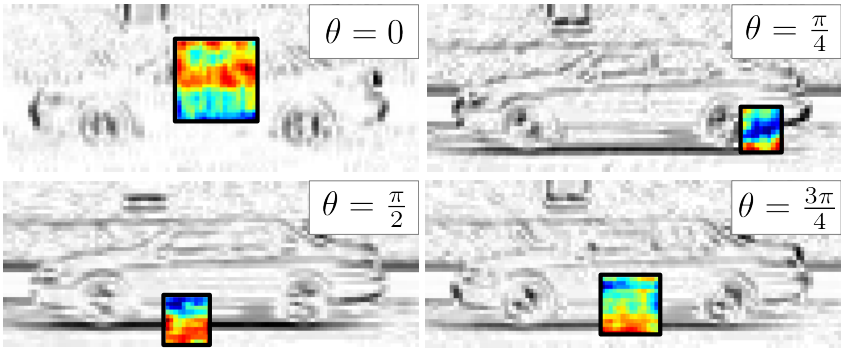
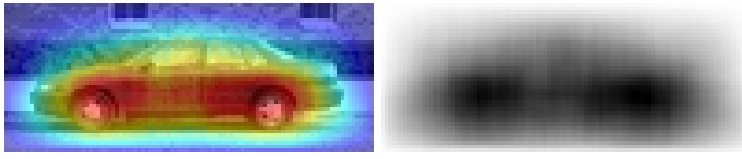


Fig. 7. Examples of filters found by our approach for the car detection task at different gradient orientations. Filters overlayed in color, gradient magnitude shown in grayscale.



Overlayed Color Grayscale

Fig. 8. Weighted filter location maps obtained for the car dataset. Each filter votes for the region $W_{\mathbf{c}_p,a_p}$ it applies on, weighted by its possible leaf values η_j and boosting coefficient α_j.

with 2000 iterations and Auto-Context was not employed, since the training data consists of patches of the exact size of the cars and it would therefore lead to severe overfitting.

Our approach outperforms all previous methods in the single-scale test set with $\mathbf{x}_\mathrm{A}$, obtaining perfect classification. Note that [20] also reported 100% accuracy in some of their runs. However, their performance on the multi-scale dataset is much lower.

In the multi-scale test set we achieve state-of-the-art performance along with [13] and [8] when employing mean-variance normalization. Note, however, that in this case we obtain a 1% improvement over [8] and [13] in the single-scale set.

Figure 7 shows examples of the filters found by our approach during the tree-building procedure. Each filter $\mathbf{k}_p$ votes for the region $W_{\mathbf{c}_p,a_p}$ it is applied on, weighted by its corresponding tree leaf values η_j and boosting coefficient α_j. It is observed that most of the filters cluster around the wheels and the car itself, which are representative parts present on car side views.

Table 2. Results on the UIUC Car Detection dataset. Performance shown as recall at recall-precision equal-error-rate, as in [8].

Method	Single-scale	Multi-scale
Xu et al. [30]†	99.5%	98%
Tivive et al. [26]†	99%	98%
Saberian et al. [24]	99.0%	92.1%
Karlinsky et al. [11]	99.5%	98.0%
Mutch et al. [20]	99.9%	90.6%
Lampert et al. [13]	98.5%	**98.6%**
Gall et al. [8]	98.5%	**98.6%**
Our approach ($\mathbf{x}_\mathrm{A}$)	**100%**	97.2%
Our approach ($\mathbf{x}_\mathrm{B}$)	99.5%	**98.6%**

† Equal precision-recall values computed approximately from the tables in [30] and [26], since the authors compare performances at the point of best F-Score instead of equal precision-recall.

6 Conclusion

We have introduced a new approach to training boosted classifiers that automatically learns features directly from image data. Our method is almost tuning-free and achieves state-of-the-art performance in a range of computer vision tasks.

In future work, we will endeavor to increase the computational efficiency of our classifiers by making the kernels separable. To this end, we will write them as products of 1D vectors and modify the training procedure accordingly.

References

1. S. Agarwal, A. Awan, and D. Roth. Learning to Detect Objects in Images via a Sparse, Part-Based Representation. *PAMI*, 2004.
2. T. Ahonen, A. Hadid, and M. Pietikïinen. Face Description with Local Binary Patterns: Application to Face Recognition. *PAMI*, 2006.
3. L. Breiman. Random Forests. *Machine Learning*, 2001.
4. R. Caruana and A. Niculescu-Mizil. An Empirical Comparison of Supervised Learning Algorithms. In *ICML*, 2006.
5. D. Cireşan, A. Giusti, L. Gambardella, and J. Schmidhuber. Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In *NIPS*, 2012.
6. N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.
7. A. Diógenes, E. Hoff, and C. Fernandes. Grain size measurement by image analysis: An application in the ceramic and in the metallic industries. In *COBEM*, 2005.
8. J. Gall and V. Lempitsky. Class-Specific Hough Forests for Object Detection. In *CVPR*, 2009.
9. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.
10. G. Hinton. Learning to Represent Visual Input. *Philosophical Transactions of the Royal Society*, 2010.

11. L. Karlinsky, M. Dinerstein, D. Harari, and S. Ullman. The Chains Model for Detecting Parts by Their Context. In *CVPR*, 2010.
12. K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning Convolutional Feature Hierarchies for Visual Recognition. In *NIPS*, 2010.
13. C. Lampert, M. Blaschko, and T. Hofmann. Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In *CVPR*, 2008.
14. M. Law and A. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *ECCV*, 2008.
15. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *PIEEE*, 1998.
16. H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *ICML*, 2009.
17. K. Levi and Y. Weiss. Learning Object Detection from a Small Number of Examples: the Importance of Good Features. In *CVPR*, 2004.
18. R. Lienhart and J. Maydt. An Extended Set of Haar-Like Features for Rapid Object Detection. In *ICIP*, 2002.
19. J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-Local Sparse Models for Image Restoration. In *ICCV*, 2009.
20. J. Mutch and D. G. Lowe. Multiclass Object Recognition with Sparse, Localized Features. In *CVPR*, 2006.
21. B. Olshausen and D. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 1997.
22. L. Quoc, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng. Building High-Level Features Using Large Scale Unsupervised Learning. In *ICML*, 2012.
23. R. Rigamonti and V. Lepetit. Accurate and Efficient Linear Structure Segmentation by Leveraging Ad Hoc Features with Learned Filters. In *MICCAI*, 2012.
24. M. J. Saberian and N. Vasconcelos. Learning optimal embedded cascades. *PAMI*, 34:2005–2018, 2012.
25. J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken. Ridge Based Vessel Segmentation in Color Images of the Retina. *TMI*, 2004.
26. F. Tivive, A. Bouzerdoum, S. Phung, and K. Iftekharuddin. Adaptive Hierarchical Architecture for Visual Recognition. *Applied optics*, 2010.
27. Z. Tu and X. Bai. Auto-Context and Its Applications to High-Level Vision Tasks and 3D Brain Image Segmentation. *PAMI*, 2009.
28. P. Viola and M. Jones. Robust Real-Time Face Detection. *IJCV*, 2004.
29. P. Wang and Q. Li. Learning Discriminant Features for Multi-View Face and Eye Detection. In *CVPR*, 2005.
30. J. Xu, Q. Wu, J. Zhang, and Z. Tang. Object Detection Based on Co-Occurrence GMuLBP Features. In *Multimedia and Expo*, 2012.
31. M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional Networks. In *CVPR*, 2010.
32. Z. Zheng, H. Zha, T. Zhang, O. Chapelle, and G. Sun. A General Boosting Method and Its Application to Learning Ranking Functions for Web Search. In *NIPS*, 2007.
33. Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng. Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. In *CVPR*, 2006.