Supervised Feature Learning for Curvilinear Structure Segmentation

Carlos Becker*, Roberto Rigamonti, Vincent Lepetit, and Pascal Fua

CVLab, École Polytechnique Fédérale de Lausanne, Switzerland
{name.surname}@epfl.ch

Abstract. We present a novel, fully-discriminative method for curvilinear structure segmentation that simultaneously learns a classifier and the features it relies on. Our approach requires almost no parameter tuning and, in the case of 2D images, removes the requirement for hand-designed features, thus freeing the practitioner from the time-consuming tasks of parameter and feature selection. Our approach relies on the Gradient Boosting framework to learn discriminative convolutional filters in closed form at each stage, and can operate on raw image pixels as well as additional data sources, such as the output of other methods like the Optimally Oriented Flux. We will show that it outperforms state-of-the-art curvilinear segmentation methods on both 2D images and 3D image stacks.

1 Introduction

Linear structures, such as blood vessels, bronchial networks, or dendritic arbors are pervasive in biological images and their modeling is critical for analysis purposes. Thus, automated delineation techniques are key to exploiting the endless streams of image data that modern imaging devices produce. Among such delineation techniques, there is a whole class of approaches such as [1–3] that take as input image segmentations in which pixels or voxels within linear structures are labeled as one and others as zero. The better the initial segmentation, the more effective these methods are. To generate them, most approaches compute a local *linearity measure* and threshold the resulting scores. This linearity measure can be postulated *a priori* [4, 5], optimized to find specific patterns [6, 7], or learned [8–10] from training data.

In this paper, we propose a novel approach to computing linearity, which yields better segmentations than using state-of-the-art methods such as [5, 10]. We introduce an original fully-discriminative method that relies on the Gradient Boosting framework [11, 12] to simultaneously compute optimal filters and boosting weights at each iteration.

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,

* This work was supported in part by the ERC grant MicroNano.

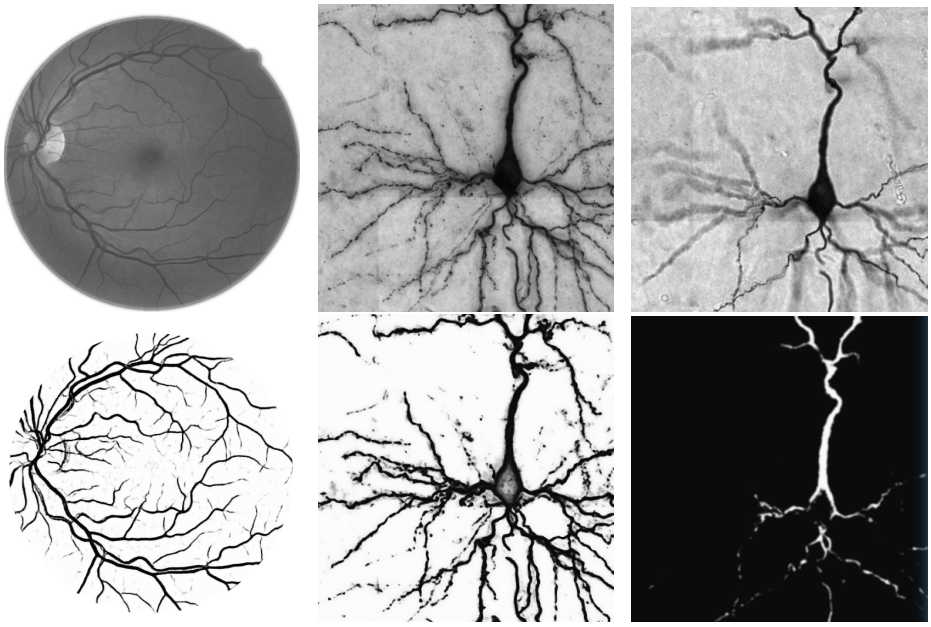

Fig. 1. Raw images **(top)** and probability maps $p(y = 1|x)$ **(bottom)** obtained with our approach on retinal scans **(left)** and brightfield microscopy in 2D **(center)** and 3D **(right)**(slice cut).

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 [13] and Deep Belief Networks (DBNs) [14] 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. The latter 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 challenging tasks [15]. However, the neural network architecture adopted in [15] requires specialized set-up and careful design, and it is computationally expensive to train, even on GPUs. By contrast, our approach is much less computationally demanding and involves just few, easily-tunable parameters. The low computational burden is particularly relevant in that it allows us to deal with 3D image stacks which, despite the prominency they are gaining in the medical domain, cannot be handled by competing algorithms such as [15] due to the high computational cost.

In the remainder of this paper, we first review briefly the standard Gradient Boosting framework, and then introduce our approach for jointly learning the filters our weak learners rely on and the parameters of the classifier that uses them. Finally, we demonstrate superior performance on several very different kinds of images that feature linear structures.

2 Gradient Boosting

Gradient Boosting [11, 12] is a an approach to approximating a function $\varphi^* : \mathbb{R}^n \to \mathbb{R}$ by a function φ of the form $\varphi(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j \, h_j(\mathbf{x})$, 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 that can make use of real-valued weak learners and minimize different loss functions [11]. Gradient Boosting has shown significant performance improvements in many classification problems with respect to classic AdaBoost [16].

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, iteratively selecting weak learners and their weights to minimize a loss function $\mathcal{L} = \sum_{i=1}^{N} L(y_i, \varphi(\mathbf{x}_i))$. We use the quadratic approximation introduced in [12]. Commonly used classification loss functions are the exponential loss $L = e^{-y_i \varphi(\mathbf{x}_i)}$ and the log loss $L = \log(1 + e^{-2 y_i \varphi(\mathbf{x}_i)})$.

The weak learners $h_j(\cdot)$ are generally either decision stumps or regression trees [11]. Regression trees are a generalization of decision stumps and usually yield significantly better performance [11], achieving state-of-the-art in many classification problems [16]. Regression trees are typically learned in a greedy manner, building them one split at a time, starting from the root [11].

3 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, a patch, or the output of a pre-processing filter —in practice we use OOF [5] for 3D data— 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.

3.1 Growing Regression Trees

Typically, Gradient Boosting implementations search through sets of weak learners that rely on a fixed set of features, such as Haar wavelets. At each iteration j, it selects the $h_j(\cdot)$ that minimizes

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

where weight-response pairs $\{w_i^j, r_i^j\}$ are computed by differentiating $L(y_i, \varphi)$ [12].

We consider here weak learners that are regression trees based on convolutions of $\mathbf{x}$ 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 denotes the tree parameters. Standard approaches learn only the θ_j, while we also learn the kernels $\mathcal{K}_j$.

Algorithm 1 Split Learning

Input: Training samples $\{\mathbf{x}_i\}_{i=1,..,N}$. Number P of kernels to explore.
Weights and responses $\{w_i, r_i\}_{i=1,..,N}$. at boosting iteration j, as in [12].
Set $\mathbb{W}$ of window locations and sizes, and set $\mathbb{L}$ of regularization factors.

// Phase I: kernel search

1: **for** $p = 1$ to P **do**

2: Pick N_{T_1} random samples from training set into T_1

3: Pick random window $W_{\mathbf{c}_p,a_p} \in \mathbb{W}$, random regularization factor $\lambda_p \in \mathbb{L}$

4: Find kernel $\mathbf{k}_p$:

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

5: **end for**

// Phase II: split search

6: Pick $\frac{N}{2}$ random samples from training set into T_2

7: **for** $p = 1$ to P **do**

8: Let $W_p(\cdot) = W_{\mathbf{c}_p,a_p}(\cdot)$

9: Find $\tau_p, \eta_{1,p}, \eta_{2,p}$ for $\mathbf{k}_p$ through exhaustive search on T_2:

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

10: Compute split cost on T_2:

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

11: **end for**

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

The tree learning procedure is performed one split at a time, as in [11]. 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

$$s(\cdot) = \begin{cases} \eta_1 \text{ if } t(\cdot) < \tau \\ \eta_2 \text{ otherwise.} \end{cases} \tag{2}$$

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

$$\sum_{i|t(\mathbf{x}_i)<\tau} w_i^j \left(r_i^j - \eta_1\right)^2 + \sum_{i|t(\mathbf{x}_i)\geq\tau} w_i^j \left(r_i^j - \eta_2\right)^2 , \tag{3}$$

where τ, η_1, and η_2 are typically found through exhaustive search [11].

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 Eq. (3) with $t(\mathbf{x}_i) = \mathbf{k}^\top \mathbf{x}_i$.

Since the space of all possible kernels is enormous, we perform this minimization in stages. Our approach is described in Alg. 1: 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 [11] 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.

3.2 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. The criterion of Eq. (1) becomes

$$\sum_{i=1}^{N} w_i^j \left(\mathbf{k}^\top W_{\mathbf{c},a}(\mathbf{x}_i) - r_i^j\right)^2 , \tag{4}$$

where $\mathbf{k}$ is now restricted to a square window parametrized by $\mathbf{c}$ and a. Given $\mathbf{c}$ and a, we can compute the optimal $\mathbf{k}$ in closed form by solving the least-squares problem of Eq. (4). To avoid overfitting, we introduce two refinements:

1. **Regularization.** We favor smooth kernels by introducing a regularization term into the criterion of Eq. (4), which becomes:

$$\sum_i w_i^j \left(\mathbf{k}^\top W_{\mathbf{c},a}(\mathbf{x}_i) - r_i^j\right)^2 + \lambda \sum_{(m,n)\in\mathcal{N}} \left(\mathbf{k}^{(m)} - \mathbf{k}^{(n)}\right)^2 , \tag{5}$$

   where $(m, n) \in \mathcal{N}$ are pairs of indexes that correspond to neighboring pixels and $\mathbf{k}^{(m)}$ is the m^{th} pixel of kernel $\mathbf{k}$. The second term in Eq. (5) imposes a smooth kernel, controlled by $\lambda \geq 0$. Note that Eq. (5) can be minimized in closed form using least squares.
2. **Splitting the training set.** Filters and splits are learned on a subset of random samples from the training set, namely T_1 and T_2, as shown in Alg. 1.

As described in Alg. 1, 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. (3). The recursive splitting procedure of Alg. 1 then produces trees that are used as weak learners in Gradient Boosting.

Note that with the exponential loss we have $r_i^j \in \{-1, 1\}$ [12]. In this case and when no regularization is imposed ($\lambda = 0$), the $\mathbf{k}$ that minimizes Eq. (4) is identical to the solution of LDA [11] 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 noisy annotations.

4 Experiments and Results

We evaluated our approach[1] on three curvilinear structure delineation tasks, considering both a 2D dataset composed by retinal scans, and a set of 3D brightfield images. The first one, called **DRIVE** [18], is a publicly-available set of 40 RGB retinal scans and 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 achieves in the segmentation task. We preserved the original splitting between train and test images. The second dataset is composed by four **brightfield micrographs of dyed neurons**. We used both the original 3D image stacks and their 2D minimum-intensity projections. The projected images are particularly complex in that both staining and projection processes introduce significant structured and unstructured noise. Also, the images are quite small compared with, for example, those of [10, 19], although the ground truth is very accurate in our case. These factors make this task particularly challenging, and therefore a good test. Furthermore, the scarcity of training samples makes learning algorithms prone to overfitting. For the 2D experiments we used two images for training and the other two for testing. For 3D we used two fully-labeled large stacks, one of size $620 \times 1300 \times 135$ for training and another one of size $768 \times 1436 \times 77$ for testing.

We compare our method against the Optimally Oriented Flux (OOF) filter [5] and [10]. [5] is a handcrafted filter, widely acknowledged as being very good for delineating tubular structures, while [10] is a hybrid approach that complements hand-crafted features with features learned in an unsupervised fashion, which outperforms [5] on both 2D datasets. The parameters of the baselines were tuned to achieve their best performance, in order to provide a fair comparison. For [10] we tried Random Forests [11] and Boosted Trees, choosing the one that yields the highest performance (Random Forests for the DRIVE dataset, Boosted Trees for the brightfield images).

In all our experiments we set $N_{T_1} = 10000$ and use the log loss to increase robustness to outliers [11], with $M = 2000$, $P = 100$ and maximum kernel size $19 \times 19 \times (19)$ 2D/(3D). We used two-level trees, $\alpha = 0.1$ [11], and $\mathbb{L} = \{500, 1000, 1500, 2000\}$ for all experiments. The only parameter varied between datasets is patch size, chosen according to the scale of the structures of interest.

Learning supervised filters in 3D is computationally intensive, so we restricted the search to symmetric separable filters, learning one component at a time with Eq. (5). This reduces training and testing time considerably. Rotation invariance in the 3D case is also a major challenge, as learning it directly from the data would require impractically many training samples. One option to solve this issue is to introduce artificial rotations of the data, but this would imply an extremely high computational burden. We therefore employed the OOF score as an additional channel, given that it incorporates the invariance we seek. This is conceptually similar to what was done in [10], where unsupervised filters are

[1] See [17] for a more detailed description of our approach and parameters. Source code available at http://cvlab.epfl.ch.

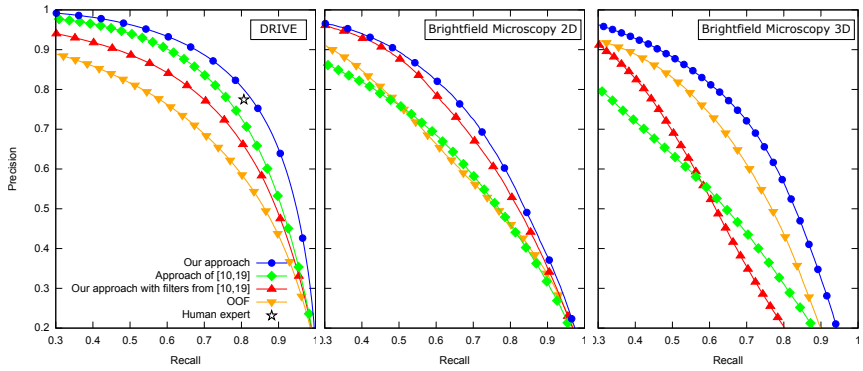

Fig. 2. (a-c) Precision-recall curves for pixel classification, obtained for different thresholds on $p(y = 1|x)$. Our approach outperforms all baselines in the 2D and 3D datasets, without the need for parameter tuning.

complemented with additional features. To reduce computational complexity further, we use stumps instead of trees for this task.

To compute the probability of each pixel in the input image of being part of a linear structure $p(y = 1|\mathbf{x})$, we take the sample vectors $\mathbf{x}$ to be image patches centered on individual pixels. Once our classifier is trained, we can compute $p(y = 1|\mathbf{x}) = (1 + e^{-2\varphi(\mathbf{x})})^{-1}$ [11]. Fig. 1 depicts some of the resulting probability maps for each dataset. Total training time was 4 hours for the 2D datasets and 6 hours for the 3D one on a 32-core 64-bit architecture.

Fig. 2 shows that our approach not only outperforms the baselines, but also outperforms human performance on DRIVE in terms of pixel error. It also yields high performance on 2D and 3D brightfield images, in spite of the inherent difficulty of this data.

To assess the importance of the supervised filter learning procedure we used the full filter banks of [10, 19], composed by 121 filters, in the context of our architecture. This was done by using the pre-computed filters as sub-patches in random positions, and then performing the usual procedure of Alg. 1. The resulting curves are labeled "*Our approach with filters from [10]*". The gap between this curve and the one for our approach shows that the supervised filter learning component of our method is largely responsible for its success.

5 Conclusion

We presented a new approach for curvilinear structure segmentation in 2D and 3D images, which automatically learns features and the classifier that uses them simultaneously. Our method outperforms current state-of-the-art curvilinear segmentation techniques, requiring almost no parameter tuning, relieving the practitioner from the time-consuming tasks of parameter and feature selection.

In future work, we will endeavor to extend our approach to textured features such as organelles in EM imagery.

References

1. Lee, T., Kashyap, R., Chu, C.: Building Skeleton Models via 3D Medial Surface Axis Thinning Algorithms. Graphical Models and Image Processing (1994)
2. Vasilkoski, Z., Stepanyants, A.: Detection of the Optimal Neuron Traces in Confocal Microscopy Images. Journal of Neuroscience Methods **178**(1) (2009)
3. Chothani, P., Mehta, V., Stepanyants, A.: Automated tracing of neurites from light microscopy stacks of images. Neuroinformatics **9**(2-3) (2011)
4. Frangi, A., Niessen, W., Vincken, K., Viergever, M.: Multiscale vessel enhancement filtering. In: MICCAI 1998. Volume 1496 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (1998)
5. Law, M., Chung, A.: Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In: ECCV. (2008)
6. Jacob, M., Unser, M.: Design of steerable filters for feature detection using canny-like criteria. Pattern Analysis and Machine Intelligence, IEEE Transactions on **26**(8) (2004)
7. Meijering, E., Jacob, M., Sarria, J.C.F., Steiner, P., Hirling, H., Unser, M.: Design and Validation of a Tool for Neurite Tracing and Analysis in Fluorescence Microscopy Images. Cytometry Part A **58A**(2) (April 2004)
8. Santamaria-Pang, A., Colbert, C., Saggau, P., Kakadiaris, I.: Automatic centerline extraction of irregular tubular structures using probability volumes from multiphoton imaging. In: MICCAI 2007. Volume 4792 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007)
9. Gonzalez, G., Aguet, F., Fleuret, F., Unser, M., Fua, P.: Steerable features for statistical 3d dendrite detection. In: MICCAI 2009. Volume 5762 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2009)
10. Rigamonti, R., Lepetit, V.: Accurate and efficient linear structure segmentation by leveraging ad hoc features with learned filters. In: MICCAI 2012. Volume 7510 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012)
11. Hastie, T., Tibshirani, R., Friedman, J.: Introduction. In: The Elements of Statistical Learning. Springer Series in Statistics. Springer New York (2009)
12. Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Sun, G.: A General Boosting Method and Its Application to Learning Ranking Functions for Web Search. In: NIPS. (2007)
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. PIEEE (1998)
14. Hinton, G.: Learning to Represent Visual Input. Philosophical Transactions of the Royal Society (2010)
15. Cireşan, D., Giusti, A., Gambardella, L., Schmidhuber, J.: Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images. In: NIPS. (2012)
16. Caruana, R., Niculescu-Mizil, A.: An Empirical Comparison of Supervised Learning Algorithms. In: ICML. (2006)
17. Becker, C.J., Rigamonti, R., Lepetit, V., Fua, P.: KernelBoost: Supervised Learning of Image Features For Classification. Technical report (2013)
18. Staal, J., Abramoff, M., Niemeijer, M., Viergever, M., van Ginneken, B.: Ridge Based Vessel Segmentation in Color Images of the Retina. TMI (2004)
19. Rigamonti, R., Sironi, A., Lepetit, V., Fua, P.: Learning Separable Filters. In: CVPR. (2013)