Convolutional Decision Trees for Feature Learning and Segmentation

Dmitry Laptev, Joachim M. Buhmann

ETH Zurich, 8092 Zurich, Switzerland

Abstract. Most computer vision and especially segmentation tasks require to extract features that represent local appearance of patches. Relevant features can be further processed by learning algorithms to infer posterior probabilities that pixels belong to an object of interest. Deep Convolutional Neural Networks (CNN) define a particularly successful class of learning algorithms for semantic segmentation, although they proved to be very slow to train even when employing special purpose hardware. We propose, for the first time, a general purpose segmentation algorithm to extract the most informative and interpretable features as convolution kernels while simultaneously building a multivariate decision tree. The algorithm trains several orders of magnitude faster than regular CNNs and achieves state of the art results in processing quality on benchmark datasets.

1 Introduction

One of the most important and critical steps for the overwhelming majority of computer vision tasks is feature design. Researchers face a challenging problem of describing local appearance of a patch around the pixel or voxel with a set of features for further processing with different machine learning algorithms.

A very important but by no means the only example of such image processing applications is medical image segmentation. The problem of constructing relevant features arises in this field in the most acute way. Experts that label pixels manually often rely only on local appearance, but are unable to mathematically define the features that appear to be most relevant for them.

In order to overcome the problem of feature design, different methods were proposed to automatically learn discriminative local descriptors [11, 13]. Among them, Deep Convolutional Neural Networks (CNN) [13] emerged as probably one of the most attractive methods for supervised feature learning nowadays. This method demonstrated to achieve superior performance for different tasks like face recognition [13], handwritten character recognition [9] and neuronal structure segmentation [7]. On the other hand, CNN suffer from the significant disadvantage that they require very large training data sets and consume an often impractical amount of time to learn the network parameters. Therefore, special hardware cluster architectures have been developed to make CNN applicable for real world tasks [7]. These constraints render the process of using CNN for end users very difficult and often even unfeasible.

This paper presents Convolutional Decision Trees (CDT): a significantly accelerated algorithm for adaptive feature learning and segmentation. It belongs to a family of oblique decision tree algorithms [10] adapted for structural data such as spatial structure of the patches in image segmentation. The algorithm builds on the following ideas:

- the method recursively builds multivariate (oblique) decision tree,
- each tree split is represented by a convolution kernel, and therefore encodes a feature of the patch around the pixel,
- convolution kernels are learned in a supervised manner while maximizing the informativeness of the split,
- regularization of kernel gradients produces interpretable and generalizable features,
- regularization parameter adaptively changes from one split to another.

These structured oblique trees significantly differ from non-structural by smoothness regularization of the learned kernels. Complexity control of feature learning renders the optimization problem more robust, regularized learning produces more interpretable features and it largely prevents overfitting. The key advantage is that the features learned adaptively for one task are informative and meaningful and, therefore, can be used for other tasks.

These ideas generate a significant performance increase compared to CNN training procedure (up to several orders of magnitude faster training), while keeping the accuracy at state of the art level. The combination of high accuracy and fast training enables anyone to use this algorithm on general purpose single processor desktop hardware.

The procedure demonstrates convincing result improvements both for medical and for natural image segmentation tasks while it avoids to employ domain-specific prior knowledge. We provide all the details in section 5. In this paper we focus on segmentation task, however, following the method described in [8], the approach can be adapted also for tasks like object detection, tracking and action recognition.

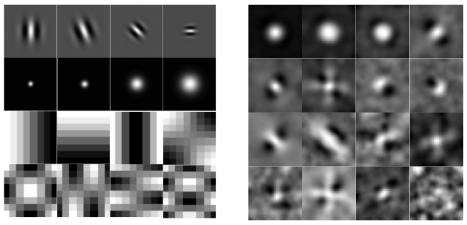

Fig. 1. Examples of different convolutional kernels: commonly used kernels (left) and the kernels obtained with the proposed algorithm (right). The algorithm finds the most informative kernels in a supervised manner, discovering meaningful kernels that look like gaussian blur, edge filters, corner and junction detectors, texture filters, etc.

2 Related work

2.1 Feature learning

The problem of feature construction is broadly discussed in the computer vision community. Standard or **commonly used features** are developed to face as many computer vision applications as possible. Different filters [1], SIFT [15] and HoG features [24] are only the few examples of such features. Even though these features work great for some tasks, they are unable to adapt to the specific problems and, therefore, often do not encode all the relevant information.

For some applications experts are able to formalize the desired properties of the object patches based on the local appearance. For such applications **domain-specific features** can be developed. Line filter transform [19] is used for blood vessels segmentation, context cue features [2] – for synapse detection. Domain-specific features proved to be very informative, however the development of these features is time-consuming and expensive while not always possible and it does not generalize to other domains. In contrast, the proposed algorithm learns features in a supervised manner and therefore adapts to the specific application without any prior information.

Unsupervised feature learning overcomes the domain-specificity, as this approach generates features based on the data itself. The "bag of visual words" representation [22] and dictionary learning [11] for sparse coding are procedures that fall in this category together with denoising autoencoders [21]. Even though these methods are powerful for data-representation, compression and image restoration, they exhibit serious limitations when applied to segmentation. This phenomenon happens because neither of the methods rely on the information about the label of the pixels and therefore learns *reconstructive*, not *discriminative* representations.

Supervised feature learning, in contrast, learns the features of the data jointly with learning the classification functions. Sparse coding algorithm can be adapted to this procedure [16], but only with classification functions limited to linear and bilinear models. Convolutional Neural Networks (CNN) [13] are able to learn more complex classification functions and more complex feature representation. CNN, as discussed in the introduction, is a very powerful and flexible technique yet with one major disadvantage: computational time for training.

For example, for neuronal segmentation dataset [6], the authors of [7] use CNN that achieves impressive accuracy, but that trains for almost a week using specially developed GPU cluster. In contrast the proposed method combines the flexibility of arbitrary convolutional kernels with the speed of decision tree training. Depending on the task first reasonable results for smaller trees can be obtained within one hour, while larger trees produce state of the art results in less than 12 hours training with one CPU which makes this method feasible for "plug-and-play" experiments. CNNs with the same training time (smaller CNNs or CNNs trained with different strategies) do not achieve comparable results. The description of the method is given in section 3.

2.2 Binary decision tree

Learning decision trees pursues the idea to consecutively split the data space into parts according to a predicate ϕ (s.t. $\phi(x) = 0$ for points x in one half-space, and $\phi(x) = 1$ for x in another half-space). The predicate is selected in such a way that it maximizes a task dependent measure of informativeness, e.g. Information Gain or Gini's diversity index [5]. $x \in \mathbb{R}^d$ is a vector of features or attributes of the object: x^j represent j-th feature of the object.

Decision trees most commonly are **univariate** [5]. Formally that means that the form of the predicate is limited to $\phi(x) = [x^j > c]$. Here $[statement]$ denotes Iverson brackets which equals to 1 if *statement* is true and zero otherwise. j and c are the parameters of the split. The choice of only univariate splits limits the computational complexity and it allows us to efficiently find the most informative split. However, it has been demonstrated that in many cases univariate trees require many more splits to learn a classifier and lead to results that are difficult to interpret [10].

Therefore **multivariate (oblique) trees** were proposed [20, 5, 10], that allow the predicate to be more flexible: $\phi(x) = [x^T\beta > c]$. Here $x^T\beta$ is a linear combination of the attributes, $\beta \in \mathbb{R}^d$ and c are the parameters of the split. Depending on the criteria of informativeness, most algorithms only return locally optimal splits.

The proposed algorithm develops the idea of oblique trees for learning convolution kernels in the context of image segmentation problems. Through regularization it incorporates the structural information about the spatial neighborhood of the pixels. Introducing this regularization helps the learned splits to be more interpretable and the optimization problem to be more robust.

3 Proposed method

Notation. As a training set we assume K pixels $i \in \{1, \dots, K\}$ with associated binary labels $y_i \in \{-1, 1\}$. Local pixel appearance is described with a patch around it. Let the size of a patch be $w \times w$, then each pixel i is represented with w^2 intensities of the pixels in the patch. In homogeneous coordinates, pixel i is described by a vector $x_i \in \mathbb{R}^{w^2+1}$ with $x_{i,1} \equiv 1$. All the vectors stored row-wise form a data-matrix $X \in \mathbb{R}^{K \times w^2+1}$, $X = [x_1, \dots, x_K]^T$.

The main idea of the method is to find a smooth convolution kernel that would be informative and discriminative for separating one class from another. A kernel of a convolution is again a $w \times w$ matrix. We also extend a vectorized kernel with a shift parameter b for the predicate. We define vector β to encodes both shift and kernel parameters: $\beta \in \mathbb{R}^{w^2+1}$, $\beta_1 \equiv b$ and $\beta_{2:w^2+1}$ encodes the kernel of the convolution.

The predicate form can be now defined as $\phi(x_i, \beta) = [\beta^T x_i > 0]$. As here we care about the sign of the convolution, we also introduce the constraint $||\beta||_2^2 = 1$ to overcome the disambiguities induced by different scalings.

3.1 Information Gain

We want to estimate the parameter vector β that would maximize the information gain $\mathrm{IG}(\beta)$. Information gain depends on the distribution of positive and negative samples before and after a split with the predicate ϕ. Let's define P as the number of all positive samples: $P = \sum_i [y_i = 1]$, $N = K - P$ — the number of negative samples. After the split, the half-space, where $\phi(x, \beta) = 1$, will contain p positive and n negative samples: $p = \sum_i [\phi(x_i, \beta) = 1, y_i = 1]$, $n = \sum_i [\phi(x_i, \beta) = 1, y_i = -1]$. Both p and n depend on the parameters β. Then

$$\mathrm{IG}(\beta) = \hat{H}\left(P, N\right) - \hat{H}_\beta\left(P, N, p, n\right), \tag{1}$$

where

$$\hat{H}\left(P, N\right) = H\left(\frac{P}{P+N}, \frac{N}{P+N}\right)$$

$$\hat{H}_\beta\left(P, N, p, n\right) = \frac{p+n}{P+N}\hat{H}\left(p, n\right) + \frac{P+N-p-n}{P+N}\hat{H}\left(P-p, N-n\right)$$

And H denotes the entropy: $H(q_0, q_1) = -q_0 \log_2 q_0 - q_1 \log_2 q_1$. Then the problem of finding the most informative split is formalized as follows:

$$\beta \in \arg\max_\beta \mathrm{IG}(\beta) \tag{2}$$

Unfortunately, the maximum of $\mathrm{IG}(\beta)$ cannot be found efficiently because of discontinuity in $\phi(x_i, \beta)$ as it contains an indicator function $[\beta^T x_i > 0]$. To overcome this issue we use an approximation from [17]:

$$\hat{\phi}_\alpha(x_i, \beta) = \frac{1}{1 + \exp(-\alpha \beta^T x_i)} \tag{3}$$

We also introduce $\hat{p}_\alpha = \sum_{i:y_i=1} \hat{\phi}_\alpha(x_i, \beta)$, $\hat{n}_\alpha = \sum_{i:y_i=-1} \hat{\phi}_\alpha(x_i, \beta)$. Then the information gain can be approximated with

$$\hat{\mathrm{IG}}_\alpha(\beta) = \hat{H}\left(P, N\right) - \hat{H}_\beta\left(P, N, \hat{p}_\alpha \hat{n}_\alpha\right)$$

It is easy to see that $\hat{\phi}_\alpha(x_i, \beta)$ converges to $\phi(x_i, \beta)$ in the limit $\alpha \to \infty$. Also $\hat{p}_\alpha$ and $\hat{n}_\alpha$ converges to p and n, respectively. This asymptotics renders it possible to solve the original problem (2) by estimating a limit process, i.e., we investigate the sequence of solutions of a relaxed problem for increasing α:

$$\beta \in \arg\max_\beta \mathrm{IG}(\beta) = \lim_{\alpha \to +\infty} \arg\max_\beta \hat{\mathrm{IG}}_\alpha(\beta)$$

3.2 Regularization

Maximizing the information gain with respect to β usually results in a split that separates the classes, but unfortunately not interpretable (see for example figure 2). More than that, when the number of training samples goes to the range of $O(w^2)$ (approximately equals to the number of parameters), the linear split

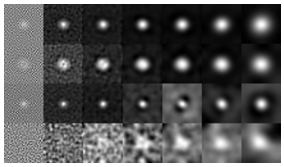

Fig. 2. Examples of convolution kernels learned with different regularization parameters λ. Each row represents a kernel from different levels of CDT (respectively from 1 to 4). Each column stands for different regularization parameters: 0.001, 0.01, 0.1, 0.5, 2, 10. Increasing regularization helps the learned features to be more interpretable (compare first columns with the last ones). However, increasing λ too much results in smoothing out relevant information (for example the orientation of the third kernel disappears in the last two columns).

model starts to overfit. This problem requires us to introduce a regularization parameter λ that penalizes the complexity of the learned kernel parameters β. We want to assure that the kernel is smooth, and, therefore, we penalize the gradient of the kernel:

$$\beta_\alpha \in \arg\max_\beta L_\alpha(\beta) \quad \text{with} \quad L_\alpha(\beta) = \hat{\mathrm{IG}}_\alpha(\beta) - \lambda||\Gamma\beta||_2^2 \tag{4}$$

Here $\Gamma \in \mathbb{R}^{2w(w-1)\ \mathrm{x}\ (w^2+1)}$ is a matrix of a 2D differentiation operator in a vectorized space, that is a Tikhonov regularization matrix.

Regularization serves two main goals. First of all, it guarantees interpretability of the kernels learned (see figures 1 and 2). And second, from an optimization point of view, a strictly concave regularization term steers the gradient descent optimization algorithm out of local minima.

3.3 Optimization

In practice, we need to choose an initial point and the gradient of the functional to effectively find a solution of the problem 4. As initial point, we use the solution to a simple regularized linear regression:

$$\beta_0 = \arg\min_\beta \frac{1}{K}||X\beta - Y||_2^2 + \lambda||\Gamma\beta||_2^2 \tag{5}$$

Here $Y = [y_1, \ldots, y_K]^T$ is a vector of all the responses and Γ denotes a Tikhonov matrix associated with the regularization above. The analytical solution to problem (5) is equal to $\beta_0 = \left(\frac{1}{K}X^TX + \lambda\Gamma^T\Gamma\right)^{-1}X^TY$.

The derivative of the functional $L_\alpha(\beta)$ in (4) can be also found analytically:

$$\frac{dL_\alpha}{d\beta} = \frac{1}{P+N}\sum_i \frac{\alpha\exp(\alpha\beta^Tx_i)}{(1+\exp(\alpha\beta^Tx_i))^2}x_i\Bigg(-\log_2\frac{p+n}{P+N-p-n} + [y_i=1]\log_2\frac{p}{P-p} + [y_i=-1]\log_2\frac{n}{N-n}\Bigg) - 2\lambda\Gamma^T\Gamma\beta \tag{6}$$

As an optimization algorithm, we employ Quasi-Newton Limited-memory BFGS (L-BFGS) [18] that estimates Hessian with low-rank approximation and, therefore, selects optimal step size.

Assuming optimization procedure as a subroutine `L-BFGS`, we sketch the algorithm that finds one split by learning the most informative convolution kernel in algorithm 1. We do not directly estimate the limit in eq. (2), but instead we iteratively increase α and initialize the optimization procedure in the next step with the solution in the previous step.

Algorithm 1 Function `findSplit` for learning the most informative split

Require: training samples $x_i, i = 1, \dots, K$ with classes y_i; λ
  $\beta_0 := \left(\frac{1}{K} X^T X + \lambda^2 \Gamma^T \Gamma\right)^{-1} X^T Y$ ▷ initialize with MSE solution
  $\beta_0 := \beta_0 / ||\beta_0||_2^2$ ▷ project on the unit sphere
  Set $\alpha := 1$
  repeat
    $\beta_\alpha := \texttt{L-BFGS}(L_\alpha(\cdot), \frac{dL_\alpha}{d\beta}(\cdot), \beta_{\alpha-1})$ ▷ find $\arg\max_\beta L_\alpha(\beta)$
    $\beta_\alpha := \beta_\alpha / ||\beta_\alpha||_2^2$ ▷ project on the unit sphere
    $\alpha := \alpha + 1$
  until $||\beta_\alpha - \beta_{\alpha-1}||_2^2 < \epsilon$ OR $\alpha >$ MaxIterations
  return β_α

4 Decision trees

How can we use the procedure `findSplit` to build a classifier? A well-known idea is to recursively split the data space into parts. The recursion stops when we achieve certainty about the label of every part. All the sequential splits end up being encoded in a binary decision tree.

The idea is very straightforward, so we do not discuss it in details, except for one important question. So far we defined the regularization parameter λ for only one split, but in principle we can change it from split to split, or from one layer of the tree to the next. Experiments show that just fixing one parameter λ to be the same for every split in a tree often results in kernel overfitting as the tree grows large.

Assume that we want to find two splits in two different parts A and B with volumes respectively $Vol(A)$ and $Vol(B)$. The relation between λ_A and λ_B for this two problems can be established from the following intuition: we want the range of both problems to be the same: for the data part $\frac{1}{K}||X\beta - Y||_2^2$ and for the regularization part $\lambda||\Gamma\beta||_2^2$. With some assumptions and derivations that are out of scope of this paper, we provide the following heuristic rule: $\lambda_B = \lambda_A \left(\frac{Vol(A)}{Vol(B)}\right)^{2/(w^2+1)}$. We approximate this ratio with just the fraction of the data points falling into each of the compacts A and B. The final recursive algorithm for building the convolutional decision tree is sketched in algorithm 2.

Algorithm 2 Function `buildTree` for decision tree construction

Require: set of indices I, λ, MaxSamples
$P := \sum_{i \in I}[y_i = 1]$; $N := \sum_{i \in I}[y_i = -1]$
treeStruct.answer $= \frac{P}{P+N}$
if $P <$ `MaxSamples` or $N <$ `MaxSamples` **then** ▷ terminal node, stop recursion
treeStruct.left $= null$; treeStruct.right $= null$
else ▷ split recoursively
$\beta := \texttt{findSplit}(x_i, y_i, \forall i \in I; \lambda)$
$I_{\text{left}} := \{i \in I : \beta^T x_i > 0\}$; $I_{\text{right}} := \{i \in I : \beta^T x_i <= 0\}$
$\lambda_{\text{left}} = \lambda \left(\frac{|I|}{|I_{\text{left}}|}\right)^{2/(w^2+1)}$; $\lambda_{\text{right}} = \lambda \left(\frac{|I|}{|I_{\text{right}}|}\right)^{2/(w^2+1)}$ ▷ change lambda
treeStruct.left $= \texttt{buildTree}(I_{\text{left}}, \lambda_{\text{left}}, \text{MaxSamples})$
treeStruct.right $= \texttt{buildTree}(I_{\text{right}}, \lambda_{\text{right}}, \text{MaxSamples})$
end if
return treeStruct

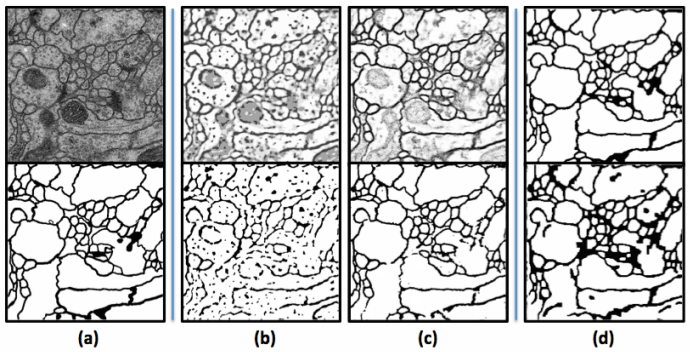

Fig. 3. The results of the proposed algorithm on Drosophila VNC dataset. Column (a) shows the input image and the ground truth. Columns (b) and (c) demonstrate the qualitative results of the algorithm for the small tree (depth = 3) and the full tree (depth = 17). The results include the probability maps (top) with a Graph Cut segmentation (bottom). The last column (d) shows the results of CNN (top) and RF with predefined features (bottom). Qualitatively the results are comparable with CNN and looks much better than RF results.

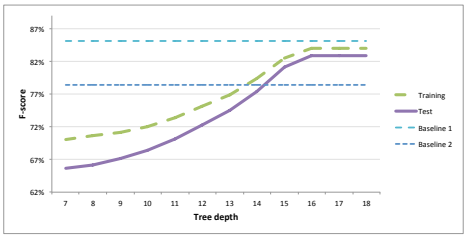

Fig. 4. Quantitative comparison. Baseline 1 is a CNN [7] which produces slightly better results, but is infeasible to train on a single CPU. Baseline 2 is a RF with Graph Cut segmentation [12], which we outperform by 4.5%.

5 Experiments

5.1 Experimental settings

We test Convolutional Decision Trees on biological and natural image datasets. First 2/3 of the images are selected for training, and accuracy is reported on the last 1/3. Because in both datasets the classes are imbalanced, we measure the accuracy in F-score: a commonly used metric that combines precision and recall.

We obtain probability maps infered by the proposed algorithm with the following parameters fixed for both datasets: $w = 31$, $\lambda = 0.5$ (initial value for the first call of `buildTree` function) and `MaxSamples` $= 50$.

All the experiments are performed on a single AMD Opteron 6174 CPU. The speed/accuracy tradeoff is controlled by the number of iterations of the `L-BFGS` subroutine. We set it in such a way that all the experiments finish within 12 hours (overnight experiment).

To get the final segmentation from the probability maps, we apply simple Graph Cut algorithm [4] with the parameters selected by 5-fold cross-validation.

5.2 Drosophila VNC dataset

As an example of a biological dataset, we use a publicly available Electron Microscopy dataset of the Drosophila first instar larva ventral nerve cord (VNC) [6]. The dataset consists of 30 images, each image depicts 2x2 microns of tissue with a resolution of 4x4 nm/pixel. The segmentation task is to annotate neuronal structures in tissue as either membranes or the inside volume of neurons.

The best results on this dataset are achieved using Convolutional Neural Networks. In terms of F-score, the accuracy of this algorithm is approximately the same as the accuracy of a human expert [7]. Our results appear to be just 2.2% worse in absolute values (82.9% CDT vs 85.1% CNN), however, the CNN training time for this dataset is around one week using GPU, comparing to just 12 hours for the proposed method. Training CNN for 12 hours produces results comparable to the other state of the art method that trains in a reasonable time and is described in [12]. A Random Forest (RF) algorithm with specially designed features produces a probability map that is then segmented with a Graph Cut algorithm that uses special potentials. Even though we use a simpler segmentation algorithm, CDT produces better probability maps. Quantitatively that results in 4.5% increase in F-score (82.9% CDT vs 78.4% RF).

5.3 Weizmann Horse dataset

The Weizmann Horse dataset [3] is well-known in the Computer Vision community. It consist of 328 manually labelled images of horses in different environments.

There are many methods that perform well on this dataset [3, 8, 23]. As a baseline we consider general purpose segmentation method based on superpixel grouping [14]. This method produces the best results on this dataset across

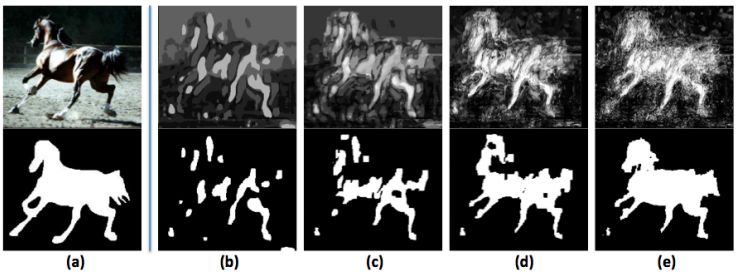

Fig. 5. The results on the Weizmann Horse dataset. Left column (a) shows the input image and the ground truth label. Each of the following columns (b)–(e) demonstrates the qualitative results of the algorithm for different tree sizes (respectively 4, 8, 12, 18). The results include the probability map (top) and the Graph Cut segmentation (bottom). Qualitatively the results improve significantly as the tree grows and more advanced features are learned.

methods that use no prior information: 79.7%. Quantitatively we achieve 80.4% and outperform it by 0.7% (insignificantly). There are also other methods that use domain-specific prior information on the shape of the horse silhouette [23] and achieve superior results of 89.2% (up to 8.8% better). However, we do not compare with them as they are limited to specific segmentation tasks where shape information is know a priori and our method is applicable to a much broader class of segmentation tasks.

6 Conclusion

In this paper we propose, for the first time, Convolutional Decision Trees: a general purpose binary segmentation algorithm that is based on learning the most informative features. We represent every feature as a convolution kernel and combine them efficiently in an oblique decision tree. We achieve interpretability and robustness by regularizing the derivative of the kernel.

The method learns features in a supervised manner and adapts to a specific problem. In this sense it works similar to Convolutional Neural Networks (CNN). The key advantage of the proposed algorithm is its run-time; it trains several orders of magnitude faster than regular CNNs which makes it possible to learn features without access to special hardware.

We test the accuracy on two benchmarks: biological (Electron Microscopy) and natural image datasets. For natural images (Weizmann Horse dataset) we achieve state of the art results across general purpose methods that require no domain-specific prior knowledge. For biological imaging (Drosophila VNC dataset) we show the results slightly inferior to CNNs, but outperform the best results that operate within a similar time period by 4.5%.

References

1. Matlab image processing toolbox, http://www.mathworks.com/help/images/
2. Becker, C., Ali, K., Knott, G., Fua, P.: Learning context cues for synapse segmentation in em volumes. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI. pp. 585–592. Springer (2012)
3. Borenstein, E., Ullman, S.: Class-specific, top-down segmentation. In: European Conference in Computer Vision – ECCV 2002, pp. 109–122. Springer (2002)
4. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. Pattern Analysis and Machine Intelligence, IEEE Transactions on 26(9), 1124–1137 (2004)
5. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. wadsworth & brooks. Monterey, CA (1984)
6. Cardona, A., Saalfeld, S., Preibisch, S., Schmid, B., Cheng, A., Pulokas, J., Tomancak, P., Hartenstein, V.: An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. PLoS biology 8(10), e1000502 (2010)
7. Ciresan, D., Giusti, A., Schmidhuber, J., et al.: Deep neural networks segment neuronal membranes in electron microscopy images. In: Advances in Neural Information Processing Systems 25. pp. 2852–2860 (2012)
8. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 33(11), 2188–2202 (2011)
9. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in Neural Information Processing Systems. pp. 545–552 (2008)
10. Heath, D., Kasif, S., Salzberg, S.: Induction of oblique decision trees (1993)
11. Kreutz-Delgado, K., Murray, J.F., Rao, B.D., Engan, K., Lee, T.W., Sejnowski, T.J.: Dictionary learning algorithms for sparse representation. Neural computation 15(2), 349–396 (2003)
12. Laptev, D., Vezhnevets, A., Dwivedi, S., Buhmann, J.M.: Anisotropic sstem image segmentation using dense correspondence across sections. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI. pp. 323–330 (2012)
13. Lawrence, S., Giles, C.L., Tsoi, A.C., Back, A.D.: Face recognition: A convolutional neural-network approach. Neural Networks, IEEE Transactions on 8(1), 98–113 (1997)
14. Levinshtein, A., Sminchisescu, C., Dickinson, S.: Optimal image and video closure by superpixel grouping. International journal of computer vision 100(1), 99–119 (2012)
15. Lowe, D.G.: Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on. vol. 2, pp. 1150–1157. IEEE (1999)
16. Mairal, J., Bach, F., Ponce, J., Sapiro, G., Zisserman, A.: Supervised dictionary learning. arXiv preprint arXiv:0809.3083 (2008)
17. Montillo, A., Tu, J., Shotton, J., Winn, J., Iglesias, J., Metaxas, D., Criminisi, A.: Entanglement and differentiable information gain maximization. In: Decision Forests for Computer Vision and Medical Image Analysis, pp. 273–293. Springer (2013)
18. Nocedal, J.: Updating quasi-newton matrices with limited storage. Mathematics of computation 35(151), 773–782 (1980)

19. Sandberg, K., Brega, M.: Segmentation of thin structures in electron micrographs using orientation fields. Journal of structural biology 157(2), 403–415 (2007)
20. Sklansky, J., Michelotti, L.: Locally trained piecewise linear classifiers. Pattern Analysis and Machine Intelligence, IEEE Transactions on (2), 101–111 (1980)
21. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning. pp. 1096–1103. ACM (2008)
22. Yang, J., Jiang, Y.G., Hauptmann, A.G., Ngo, C.W.: Evaluating bag-of-visual-words representations in scene classification. In: Proceedings of the international workshop on Workshop on multimedia information retrieval. pp. 197–206. ACM (2007)
23. Zhu, L., Chen, Y., Yuille, A.: Learning a hierarchical deformable template for rapid deformable object parsing. Pattern Analysis and Machine Intelligence, IEEE Transactions on 32(6), 1029–1043 (2010)
24. Zhu, Q., Yeh, M.C., Cheng, K.T., Avidan, S.: Fast human detection using a cascade of histograms of oriented gradients. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. vol. 2, pp. 1491–1498. IEEE (2006)