High Entropy Ensembles for Holistic Figure-ground Segmentation

Ignazio Gallo
ignazio.gallo@uninsubria.it
Alessandro Zamberletti
a.zamberletti@uninsubria.it
Simone Albertini
simone.albertini@uninsubria.it
Lucia Noce
lucia.noce@uninsubria.it

Applied Recognition Technology Laboratory
Department of Theoretical and Applied Science
University of Insubria
Varese, Italy

Abstract

Modern computer vision applications are built on processing pipelines with hard and intensive tasks; among those, figure-ground segmentation is definitely one of the most important and challenging. As proved by many works in literature, an effective way of approaching this task consists in combining different algorithms in structures where they collaborate towards the solution of the given segmentation problems. Inspired by other model combination frameworks, we propose a novel method to create tree-based ensembles of randomly configured figure-ground segmentation algorithms. The tree-based topology enables the algorithms to communicate with each other to let the strengths of one overcome the weaknesses of the others and vice versa, while the randomness injection reduces the risk of overfitting, decreases the computational complexity of the model creation procedure and enables our ensembles to overcome state-of-the-art results for several challenging datasets.

1 Introduction

Many works in literature have proved that one of the most effective method of dealing with the task of figure-ground segmentation [26] consists in combining different segmentation algorithms together [11, 19, 38]. For example, boosting procedures to automatically build a set of weak classifiers [12, 29] have been successfully used as underlying learning technique for addressing different object identification and segmentation tasks [33, 37]. Excellent results were also obtained by employing other model combination methods, such as cascaded classifier models [16], bayesian model averaging [17], greedy trees, multiple kernels [34], cascade of boosted ensembles [28, 32, 35], parallel strong learners [1] or decision forests [8, 14, 15, 30] to name a few.

Several among these model combination frameworks adopt rejection rules to improve the classification time of the ensembles at the cost of reducing the interactions between different elements in the structures; however, as proved by CCM [16], when execution time

is not critical, a joint approach that focuses on interaction between classifiers can be extremely effective. While CCM focuses on combining algorithms for solving different tasks (object detection, region labeling, geometric reasoning, etc), a similar approach could lead to good results even when combining algorithms for solving a single task. In fact, considering the task of figure-ground segmentation, even though different segmentation algorithms may obtain similar overall results for a given dataset, they may commit substantially different prediction errors for the same set of given samples [19], therefore good results can be expected when appropriately combining those algorithms together.

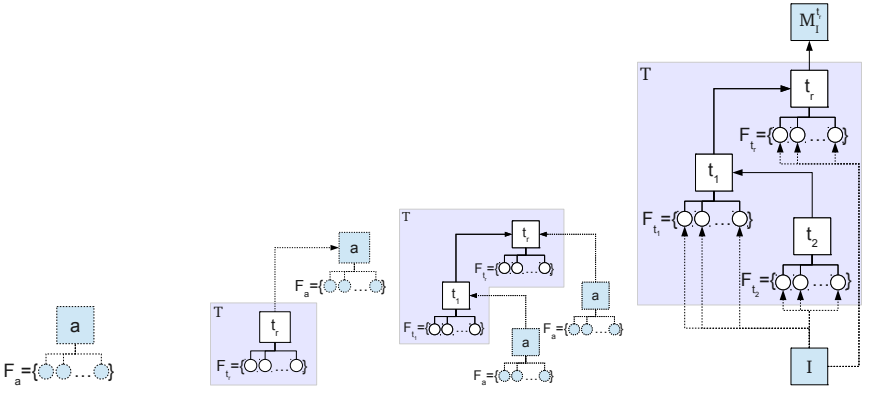
Following this idea, we propose a novel model combination framework for combining a set of figure-ground segmentation algorithms into dynamically built High Entropy Ensembles (HEE) whose creation phases are guided by the maximization of a goodness function. Our method is fully automatic and requires no user input. Unlike most of the model combination approaches proposed in literature [1, 16, 32], we do not focus on looking for the optimal classifiers to be added to the HEEs, instead we pick them from a pool of randomly configured segmentation algorithms. Similar suboptimal approaches have already been successfully employed by other works in literature [8, 9, 15]; however, we push this randomization concept even further to prove that state-of-the-art results can not only be achieved but also be overcome by extremely randomized algorithms appropriately combined into chaotic highly connected ensembles.

In Sec. 2 the proposed model is introduced, while in Sec. 3 an extensive experimental analysis is conducted over HEEs generated from a pool of two significant figure-ground segmentation algorithms. We provide comparisons with some of the most popular model combination frameworks and figure-ground segmentation methods, we evaluate our tree-based HEEs against equivalent linear cascades and we aggregate them to form forests. The proposed ensembles are able to outperform recent state-of-the-art approaches for different challenging segmentation datasets: INRIA Graz-02 [22], Weizmann Horses [3], Oxford Flower 17 [23] and the single class segmentation variant of VOC2010 [10, 20].

2 Proposed model

The proposed method consists in a *building phase* that automatically generates a figure-ground segmentation model by combining a set of randomly configured figure-ground segmentation algorithms.

Given a dataset D of images, we build the validation set V by randomly selecting one third of the images in the training set of D. Let $\mathbb{A}$ be a pool of figure-ground segmentation algorithms and $\mathbb{F}$ be a set of feature extractors. Each $a \in \mathbb{A}$ must satisfy the following constraints: (i) it produces as output a single gray-scale image in which the intensity level assigned to each pixel represents the probability that it belongs to foreground, (ii) it requires as input a set of patterns generated by a set of feature extractors $F_a \subset \mathbb{F}$ or gray-scale images produced by other algorithms in $\mathbb{A}$, similarly to [16]. In Sec. 2.1 and 2.2 we provide a detailed description of the figure-ground segmentation algorithms $\mathbb{A}$ and feature extractors $\mathbb{F}$ used to obtain the experimental results of Sec. 3. The *building phase* builds a segmentation tree T by combining a set of randomly configured algorithms from $\mathbb{A}$ (called nodes). The whole method is driven by the maximization of a goodness function $G(T,V)$ computed as the VOC average accuracy [10] of T for the validation set $V \subset D$. We define G as such to take into account pixel-wise precision and recall. The segmentation map M_I for an image I is generated by T by giving I as input to all the feature extractors $f \in F_{t_i}\ \forall t_i \in T$. The feature



(a) base step (b) bottom-up step (c) top-down step (d) segmentation of I

Figure 1: An example showing how the *building phase* creates a tree T: the base step (a) creates the first node of T, the bottom-up step (b) adds a new root node to T, the top-down step (c) adds a new leaf node to T. Once the *building phase* ends, it is possible to let T produce the segmentation map of an image I by giving I as input to all the feature extractors in T (d).

patterns generated by the extractors in $F_{t_i} \subset \mathbb{F}$ are exploited by the node $t_i \in T$ to produce a segmentation map $M_I^{t_i}$ that is given as input to its parent. This procedure is repeated until a segmentation map $M_I^{t_r}$ is produced by the root node $t_r \in T$. The final binary segmentation map M_I is obtained by thresholding $M_I^{t_r}$ with a threshold level equals to half the maximum intensity value of the map. Note that, even though the ensembles built by our combination approach are trees, we graphically represent them as directed acyclic graphs to emphasize that the segmentation process starts from the leaves and ends at the root node.

The first step of the *building phase* of T is called *base step*.

Base step - the goal of this step is to create the first node of the tree T, as shown in Fig. 1a. We randomly select a figure-ground segmentation algorithm $a \in \mathbb{A}$ along with a random set of feature extractors $F_a \subset \mathbb{F}$. After that, we train the selected node a using the patterns produced by the feature extractors in F_a. We repeat the previous actions n times until we identify $\bar{a}$ as the node that maximizes $G(a,V)$ over all the n generated nodes. Finally, $\bar{a}$ becomes the first and only node $t_r \in T$.

In order to extend T, after the *base step* we perform a *bottom-up step* followed by a *top-down step*.

Bottom-up step - the goal of this step is to add a new root node to T, as shown in Fig. 1b. Similarly to the *base step*, we perform a random selection of a figure-ground segmentation algorithm $a \in \mathbb{A}$ along with a random set of feature extractors $F_a \subset \mathbb{F}$. Then, we create a new tree $\tilde{T}$ by adding a as parent of the root node $t_r \in T$. This means that the output produced by the root node t_r is given as input to a along with the set of patterns generated by the feature extractors in F_a. We train a using the patterns produced by its feature extractors in F_a and the gray-scale map generated by t_r. We repeat the previous actions n times, always starting from the original tree T. Finally, we identify $\bar{T}$ as the tree $\tilde{T}$ that maximizes $G(\tilde{T},V)$ over all the n generated trees. We replace the original tree T with $\bar{T}$ if and only if $G(\bar{T},V) - G(T,V) > \varepsilon$.

Top-down step - the goal of this step is to add a child node to each node $t \in T$, as shown

in Fig. 1c. We create a new tree $\bar{T}$ by adding a as a child of t. This means that the output produced by a is given as input to t along with the set of patterns generated by the feature extractor algorithms in $F_t \subset \mathbb{F}$. We recursively retrain the nodes of $\bar{T}$ starting from a all the way up to the root. We repeat the previous actions at most n times (always starting from the original tree T) until $G(\bar{T},V) - G(T,V) > \varepsilon$. If this condition is satisfied then $\bar{T}$ becomes T. Note that we iterate the procedure only over the nodes that belong to the starting tree T.

The two previous steps are recursively executed until a *bottom-up step* followed by *top-down step* do not add any new nodes to T. The final ensemble can be used to produce the segmentation map for a given image I as summarized in Fig. 1d. The only non-randomly chosen parameters of our model are n and ε; the others are chosen inside a set of coherent values during the random selection performed at each step. This is not new as other works obtain good results by injecting randomness in their models [8, 15]. Our method is far from boosting based approaches [35, 37] or CoBEs [28, 32] because the interaction between a node and its children in T is much different from the interaction between a set weak learners in a strong learner and also because we do not employ rejection rules. In fact, while the output of a strong learner is usually obtained by computing a weighted sum over the predictions produced by its weak learners [12, 29], the one produced by an internal node $t \in T$ does not only depend on the patterns generated by the feature extractors in F_t but also on the outputs produced by its children, similarly to CCM [16]. The choice of using a top-down step in addition to a classic bottom-up selection (as in common cascade combination models) is because, by adding new leaf nodes at the base of the tree, we enrich the ensemble with information that is close to the input image I; this acts as a regularization factor as it increases the generalization ability of the structure, helps prevent getting stuck at local maxima of the goodness function and reduces the risk of overfitting, as proved in Sec. 3.1. We chose to use tree-based models because, as investigated in [11, 38] and Sec. 3.2, they usually perform better than similar multi-response linear structures. Moreover, at each level inside a tree we can combine multiple weak and strong learners, while in a cascade this is not permitted, as in CoBEs, CCM, model averaging and other linear model combination methods.

2.1 Figure-ground segmentation algorithms

We perform our experiments using two figure-ground segmentation algorithms: a sliding window MLP [27] network and a model inspired by the one of [21]. In Sec. 3 they are called *Sw* and *Seg* respectively. They both share the output image size s as common parameter. Similarly to [16], the patterns generated by each segmentation algorithm in $\mathbb{A}$ are represented as images, where each pixel denotes the probability of a pixel in the original image scaled by s belonging to either foreground or background.

In computer vision, sliding window approaches are frequently employed to analyze the content of an image. Several works assign a classification score to image sub-windows by computing a set of features within these regions [6, 35] and exploit this information to accomplish their tasks. In our experiments we employ a sliding window paired with a MLP network to assign a score to each pixel of a given image. The MLP network is composed by two hidden layers whose size is equal to half the dimension of the input layer. It receives as input the values produced by a sliding window that reads the patterns generated by both a set of feature extractors or other figure-ground segmentation algorithms. The size of the sliding window w is randomly chosen during the *building phase*. The model is trained using the *rprop* algorithm with the default configuration from Igel *et al.* [18].

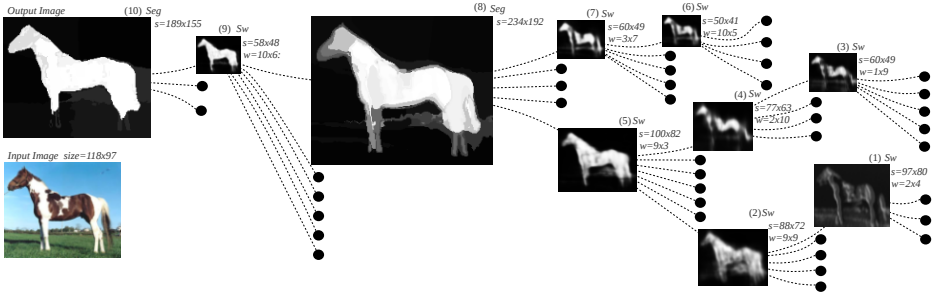


Figure 2: An example showing the segmentation tree that outperforms the state-of-the-art results for the Weizmann Horses dataset. The different nodes compensate each others' errors: *Seg* nodes produce accurate segmentations near the contours but lose parts of the object, *Sw* nodes produce blurred segmentations but capture the whole object. The segmentation map for the given input image is produced in less than one second.

The *Seg* algorithm is similar to the model of [21]; to speed it up we modified CPMC [4] by segmenting the given image using k-means. The number of centroids k is randomly chosen at each selection performed during the *building phase*. Similarly to [21], each segment is assigned a figure-ground score based on the input patterns; the intensity level assigned to each segment in the output soft segmentation map depends on its score.

Even though it would be interesting to include a higher number of algorithms in $\mathbb{A}$, by using two simple models that overcome other state-of-the-art methods when combined by our framework we better highlight the effectiveness of our approach.

2.2 Feature extractors

In our experiments we employ a set of feature extractor algorithms, each of them receives as input an image I and produces a pattern p_I by computing one feature from the following sets: (i) the filters bank proposed by Winn *et al.* [36], based on the perceptually uniform *CIE Lab* color space and consisting of scaled Gaussians, xy derivatives of Gaussians and Laplacians of Gaussian, (ii) the Haar-like features proposed by Papageorgiou *et al.* [24], (iii) the gradient filters described by Dalal and Triggs [6].

3 Experiments

In this section we present the results obtained by performing an extensive experimental analysis of the proposed method. Our results prove that: (i) the algorithm presented in Sec. 2 effectively builds an ensemble whose nodes collaborate towards the solution of the given segmentation task (Sec. 3.1), (ii) the segmentation trees produced by our method can outperform state-of-the-art approaches for some of the commonly used figure-ground segmentation datasets (Sec. 3.2).

3.1 Building phase analysis

Here we prove the effectiveness of the proposed approach by showing that: (i) randomly configured *Sw* and *Seg* nodes achieve poor results when used individually, (ii) better perfor-

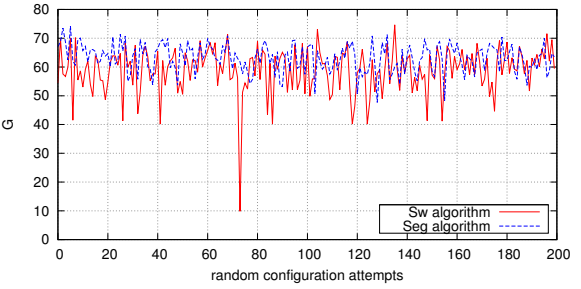


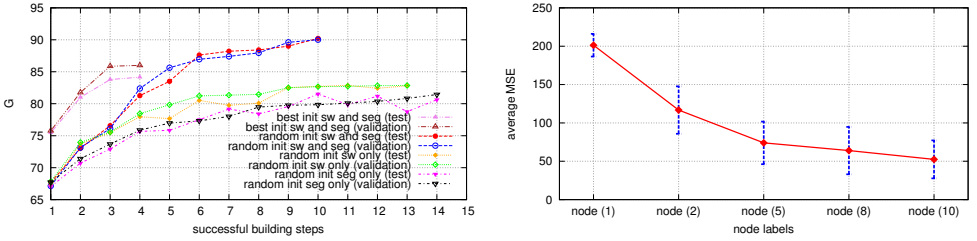
Figure 3: Goodness value G for the Weizmann Horses test set when using single nodes generated by randomly configuring 200 times the algorithms *Sw* and *Seg*. They yield similar poor results when used independently but commit different prediction errors, as such they are able to collaborate effectively when combined together, as shown in Fig. 4a.

mances can be obtained if we combine some of these nodes in a tree T, (iii) the information produced by each node of T is positively exploited by its parent. In all the experiments we set $n = 12$ and $\varepsilon = 0.05$. When not explicitly stated otherwise, we use $\mathbb{A}$ and $\mathbb{F}$ as described in Sec. 2.1 and 2.2. The parameters for the algorithms in $\mathbb{A}$ are randomly chosen in the following ranges: $s \in [10,100]$, $w \in [1,10]$ and $k \in [5,50]$. To reduce the computational complexity of the nodes we set the following constraint: $|F_t| \leq 10 \; \forall t \in T$, meaning that one node may have at most 10 input feature extractors. In our experiments, the above parameter configuration provided the best compromise between final overall results and computational cost of the *building-phase*. We perform all the experiments of this section on the Weizmann Horses dataset.

First, we start by generating a set of nodes by randomly configuring 200 times each algorithm in $\mathbb{A}$; after that, we compute the goodness values they individually achieve on test set. The results we obtain are shown in Fig. 3. We observe that the randomly generated nodes yield poor mean results μ when used individually and that the standard deviation σ of their goodness values is small: $\mu_{Sw} = 58.90$, $\sigma_{Sw} = 7.87$, $\mu_{Seg} = 63.13$, $\sigma_{Seg} = 5.95$. To prove that it is possible to obtain better results if some of these nodes are combined using our approach, we perform a second experiment in which we execute the *building phase* to generate a tree T; after each successful step we report the goodness of T for the validation and the test sets. Results are shown in Fig. 4a. Taking into account that each successful step adds a new node to T, we observe that the goodness of T on validation increases according to the ensemble's size and that the number of algorithms in $\mathbb{A}$ deeply affects the performances of T.

The first observation is not surprising, since every successful step always increases the goodness of a tree on validation (as described in Sec. 2). The latest is more interesting, because it is in accordance with the assumption that different segmentation algorithms may commit different mistakes [19] and it proves that the algorithms of $\mathbb{A}$ collaborate in T so that the errors committed by one are compensated by the others and vice versa. This behavior can be observed in Fig. 4a, where the tree obtained using $\mathbb{A} = \{Sw, Seg\}$ outperforms the ones obtained using $\mathbb{A} = \{Sw\}$ and $\mathbb{A} = \{Seg\}$ after the third successful step.

An additional experiment is conducted to verify whether the correct information produced by each node of T is positively exploited by its parent. It consists in computing the average Mean Square Error (MSE) produced by the nodes of the tree of Fig. 2 on the test set. The obtained results are presented in Fig. 4b. To make the graph more readable, we compute



(a) (b)

Figure 4: (a) Comparison between goodness achieved for the Weizmann Horses test set after each successful step of the *building phase* while varying the algorithms in $\mathbb{A}$ and the parameters selection technique. The *building phase* for the mixed *Sw* and *Seg* tree terminates after just 10 steps and achieves significantly better results both on test and validation than the ones obtained using an optimal parameter initialization approach (*best init*) or just one type of segmentation algorithm. (b) Average mean square error achieved for the Weizmann Horses test set by the nodes of the model showed in Fig. 2, on a path from a leaf to the root. The reported error bars represent the standard deviation of the mean square error. The error decreases as we climb the ensemble toward its top node.

the average MSE values only for the nodes lying on the path that goes from the leaf node labeled in Fig. 2 as (1) to the root (10). Results show that the mean square error decreases as we move from leaf nodes towards the root and this proves that some of the correct predictions produced by nodes sharing the same level inside the tree are correctly transferred to their ancestors. As we move towards the root node (10), the improvement in terms of average MSE becomes less significant because we are approaching the optimal solution.

We perform another experiment in which we provide a comparison between the random and the fixed parameters selection approaches (*best init*). In the fixed parameters selection approach we perform a grid search to compute the parameters that maximize the goodness value for each of the algorithms in $\mathbb{A}$ and we force the *building phase* to create a tree composed only by optimal nodes, similarly to the linear models of [9, 16]. The grid search is performed in $s \in \{10, 100\}$, $w \in \{1, 10\}$ and $\mathcal{P}(\mathbb{F})$, randomly taking 20 samples for each dimension. Results are shown in Fig. 4a. We observe that the *building phase* for the *best init* approach terminates after adding just 4 nodes to the ensemble. This does not surprise since, by exhaustively looking for the best node at each step, we approximate other model combination frameworks (such as CCM or PSL); the ensembles generated by those methods are usually not deep, mostly to prevent over-fitting the training data.

Additional experiments were performed to determine whether all the nodes in the trees contribute toward the final predictions. We experienced a significant drop in performance when we recursively removed leaf nodes without retraining the nodes in the trees not affected by the change; the same behavior was observed when removing entire branches. However, we believe it is possible to define a pruning procedure that coherently removes nodes to speed up the segmentation process at the cost of reducing overall performances.

It is particularly interesting to observe that in many cases the set of image features automatically selected by the *building phase* as input features for the nodes in the HEE resembles the base set of Integral Channel Features [7] (LUV, gradient histogram and gradient magnitude) widely used by state-of-the-art rigid object detection algorithms. This proves that, even

(a) positive (b) negative

Figure 5: Examples of positive and negative segmentation results for images from different classes of INRIA Graz-02 dataset. $M_I^{t_r}$ denotes the soft figure-ground segmentation maps generated by the root node t_r of HEE for the given input image I, while M_I represents the hard binary mask obtained by thresholding $M_I^{t_r}$ to half its maximum intensity value. M_I is superimposed over I for visualization purposes.

tough the proposed model is heavily random-based, it tries to build optimal segmentation ensembles.

The time required to complete each successful step of the *building phase* for the model presented in Fig. 2 is exponential in the number of previous successful steps. In fact, as we approach an optimal solution, it becomes difficult to find nodes that further increase the goodness of the tree being built. Due to the non deterministic nature of the *building phase*, we cannot determine a priori how long it takes to complete its execution. The *building phase* for the model of Fig. 2 lasted approximately 3 days. The time required by a tree to perform the segmentation of an image depends on the computational complexity of its nodes; the model of Fig. 2 processes a 150×150 pixels image in less than one second.

3.2 Results

In this section, we compare the ensembles built using the proposed model with other state-of-the-art algorithms and model combination methods. For all the experiments, we use the same set of parameters described in Sec. 3.1. The comparisons are carried out for the following datasets: Weizmann Horses [3], Oxford Flower 17 [23], INRIA Graz-02 [22] and the f/g variant of VOC2010 [20].

The segmentation performances for the first two datasets were measured using the same metrics of [2]. Results for the INRIA Graz-02 dataset were measured using F-measure, we did not employ the PRC equal error rates for the same reason of [20]. For the figure-ground variant of the VOC2010 dataset, we used the classic pixel-wise intersection-over-union (IoU) as in [20]. In each experiment, we provided comparisons with AdaBoost [12], bayesian averaging [17], CCM [16] and CoBE [32, 35]. For the Weizmann Horses dataset, we also compared with the auto-context-cascade [33], which is a boosted cascade of auto-context classifiers.

Table 1: Comparisons with other state-of-the-art methods.

(a) Weizmann Horses

Method	S_a (%)	S_o (%)
[20]	94.7	/
[4]	94.6	80.1
[31]	95.4	/
[12]	90.0	72.9
[16]	89.3	79.6
[17]	77.1	58.9
[35]	90.8	76.0
LHEE	87.1	72.5
HEE	98.2	90.2

(b) Oxford Flower 17

Method	S_a (%)	S_o (%)
[23]	/	94.0
[4]	97.7	92.3
[5]	/	90.4
[12]	93.1	85.5
[16]	86.3	84.5
[17]	87.3	81.0
[35]	95,8	90,6
LHEE	89,6	87,6
HEE	98.1	96.1

(c) INRIA Graz-02

Method	cars	people	bikes	avg.
[22]	53.8	44.1	61.8	53.2
[20]	74.8	66.4	63.2	68.1
[13]	72.2	66.3	72.2	70.2
[12]	60.1	48.6	63.0	57.2
[16]	62.6	55.9	72.8	64.4
[17]	55.4	53.4	65.3	56.0
[35]	75.4	67.0	73.8	72.1
LHEE	66,7	54,9	72,1	64.6
HEE	82.4	67.9	78.2	76.2

(d) VOC2010

Method	IoU %
[20]	48
[4]	34
[25]	46
HEE	56

AdaBoost was applied to the algorithms in $\mathbb{A} = \{Sw, Seg\}$ in the following way: we built a family of figure-ground segmentators $\mathbb{H}$ using the same process for selecting the best node during the *base init*, selecting at most 20 optimal $h_t \in \mathbb{H}$ by computing the weight distribution D_t over the pixels in the training set. Similarly, we obtain the results for CoBE (nodes are boosted using AdaBoost as in [35]) and the model averaging approaches. In our experiments, we adopted the CCM framework for solving the task of figure-ground segmentation, treating the algorithms in $\mathbb{A}$ as *black boxes*. However, since CCM does not provide a method for combining more than one type of segmentation algorithm in the same structure, we built a cascade for each one of the two algorithms in $\mathbb{A}$. As in the original paper, we used cascades of fixed sizes 2 and 5 and we reported the best results among those achieved by the four 2-CCM and 5-CCM cascades.

To prove whether tree structures perform better than equivalent linear cascades, we modified our method to build degenerate trees without rejection rules (LHEE). Unsurprisingly, the proposed tree-based structures always performed better than equivalent cascades, as expected from the results of [11, 38]. We have also built forests of sizes 20 and 5 of HEEs (max depth 6 and 20 respectively, using the same set of features of [30]), in which the final predictions were obtained by averaging the outputs of all the HEEs. The obtained results were always close but lower than those of single HEEs.

The horizontal lines in Table 1 separate standard segmentation algorithms from model

combination frameworks.

Weizmann Horses - We use 5-fold cross-validation. Results are shown in Table 1a. The HEE of Fig. 2 outperforms all the other methods and the auto-context-cascade [33], obtaining an F-measure of 0.91. It also outperforms the recent Cascaded Hierarchical Model [31] and the Hough Forest method of [14].
Oxford Flower 17 - We only consider the subset of 848 images labeled for segmentation. Results are reported in Table 1b. Tree-based HEEs outperform cascade HEEs and the other model combination frameworks. LHEEs perform even worse than CoBE [35].
INRIA Graz-02 - Similarly to other methods [13, 20, 22], we build an independent model for each class using the first 150 odd-numbered images for training and the first 150 even-numbered for test. In Table 1c we report the obtained accuracies; the poor results for the *people* class are due to the large number of possible variations in articulations and poses. As suggested in [22], the use of a larger training set may improve the overall results. Examples of positive and negative segmentation results for this dataset are presented in Fig. 5, they highlight the limits of the output hardening function in generating the final segmentation maps.
Figure-ground single class VOC2010 - As in other figure-ground segmentation works, we fuse the ground-truth maps for the 20 object classes into single foreground/background maps. We test using the 964 images in the validation set and train on the original public training set. As shown in Table 1d, HEE obtains 56.1 IoU%, which is considerably higher than other recent methods [4, 20, 25].

4 Conclusion

The effectiveness of the proposed model combination framework has been proved by the results of an extensive experimental analysis conducted on both the model creation procedure and the final ensembles. It is surprising how such a simple framework that requires no user input nor extensive tuning constitutes a valid alternative to other widely used model combination methods when combining heterogeneous segmentation algorithms. It is an open question whether our method can pose a challenge to other similar approaches when applied to more challenging tasks, such as object classification or multi-class image segmentation. In future works, it would be interesting to cast it in a more general principled framework in which algorithms for solving different tasks can be combined together to overcome their individual weaknesses.

References

[1] Andre Barczak, Martin Johnson, and Chris Messom. Empirical evaluation of a new structure for adaboost. In *Symposium On Applied Computing*, 2008.

[2] Luca Bertelli, Tianli Yu, Diem Vu, and Burak Gokturk. Kernelized structural SVM learning for supervised object segmentation. In *International Conference on Computer Vision and Pattern Recognition*, 2011.

[3] Eran Borenstein, Eitan Sharon, and Shimon Ullman. Combining top-down and bottom-

up segmentation. In *International Conference on Computer Vision and Pattern Recognition*, 2004.

[4] Joao Carreira and Cristian Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2012.

[5] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. BiCos: A bi-level co-segmentation method for image classification. In *European Conference on Computer Vision*, 2011.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision and Pattern Recognition*, 2005.

[7] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *British Machine Vision Conference*, 2009.

[8] Piotr Dollár and Larry Zitnick. Structured forests for fast edge detection. In *International Conference on Computer Vision*, 2013.

[9] Piotr Dollár, Peter Welinder, and Pietro Perona. Cascaded pose regression. In *International Conference on Computer Vision and Pattern Recognition*, 2010.

[10] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[11] Eibe Frank, Yong Wang, Stuart Inglis, Geoffrey Holmes, and Ian H. Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998.

[12] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55 (1):119–139, 1997.

[13] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision*, 2009.

[14] Juergen Gall and Victor Lempitsky. Class-specific hough forests for object detection. In *International Conference on Computer Vision and Pattern Recognition*, 2009.

[15] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[16] Geremy Heitz, Stephen Gould, Ashutosh Saxena, and Daphne Koller. Cascaded classification models: Combining models for holistic scene understanding. In *International Conference on Neural Information Processing Systems*, 2008.

[17] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4), 1999.

[18] Christian Igel, Marc Toussaint, and Wan Weishui. *Rprop using the natural gradient*. 2005.

[19] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[20] Daniel Küttel and Vittorio Ferrari. Figure-ground segmentation by transferring window masks. In *International Conference on Computer Vision and Pattern Recognition*, 2012.

[21] Fuxin Li, Joao Carreira, and Cristian Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *International Conference on Computer Vision and Pattern Recognition*, 2010.

[22] Marcin Marszałek and Cordelia Schmid. Accurate object localization with shape masks. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[23] Maria Elena Nilsback and Andrew Zisserman. Delving deeper into the whorl of flower segmentation. *Image and Vision Computing*, 28(6):1049–1062, 2010.

[24] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *International Conference on Computer Vision*, 1998.

[25] Amir Rosenfeld and Daphna Weinshall. Extracting foreground masks towards object recognition. In *International Conference on Computer Vision*, 2011.

[26] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3): 309–314, 2004.

[27] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning internal representations by error propagation*. 1986.

[28] Mohammad J. Saberian and Nuno Vasconcelos. Boosting classiiňĄer cascades. In *International Conference on Neural Information Processing Systems*, 2010.

[29] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[30] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *British Machine Vision Conference*, 2008.

[31] Mojtaba Seyedhosseini, Mehdi Sajjadi, , and Tolga Tasdizen. Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks. In *International Conference on Computer Vision*, 2013.

[32] Teo Susnjak, Andre L. C. Barczak, and Ken A. Hawick. Adaptive cascade of boosted ensembles for face detection in concept drift. *Neural Computing and Applications*, 21 (4), 2012.

[33] Zhuowen Tu and Xiang Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1744–1757, 2010.

[34] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *International Conference on Computer Vision*, 2009.

[35] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2), 2004.

[36] John Winn, Antonio Criminisi, and Thomas Minka. Object categorization by learned universal visual dictionary. In *International Conference on Computer Vision*, 2005.

[37] Bo Wu and Ramakant Nevatia. Simultaneous object detection and segmentation by boosting local shape feature based classifier. In *International Conference on Computer Vision and Pattern Recognition*, 2007.

[38] Bernard Zenko. Is combining classifiers better than selecting the best one? *Machine Learning*, 54(2):255–273, 2004.