Università degli Studi dell'Insubria
DiSTA - Dipartimento di Scienze Teoriche e Applicate
Ph.D Course in Computer Science
XXVII cycle of study

Ensembles of Segmentation Algorithms for Figure-ground Segmentation

Ph.D Thesis
SIMONE ALBERTINI

Advisor: IGNAZIO GALLO

Year 2014

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deep gratitude to my advisor, Professor Ignazio Gallo and all the Applied Recognition Technology Laboratory (ArteLab) present and former members which helped me during my Ph.D course: Angelo Nodari, Marco Vanetti, Alessandro Zamberletti and Lucia Noce. My thanks also go to Professor Pierluigi Gallo who revised this thesis.

I will also like to thank the company that funded my studies, 7pixel, and especially its CEO Nicola Lamberti and Andrea Broglia for supporting the ArteLab's activities by providing funds and places to perform our researches and for always believing in scientific research as an investment for the future.

My special thanks also go to all my co-workers in 7pixel Varese: they are a lot now and even if I cannot name them all, I would like to express my gratitude for contributing to build an exciting working environment.

Last but not least, I would like to thank my parents Maurizio and Pinuccia and my girlfriend Sara for encouraging and believing in me throughout my studies.

Olgiate Olona, December 2014

List of Symbols

T	Ensemble of algorithms structured as a tree, both MNOS and HEE.
Sw	Multilayer Perceptron Neural Network algorithm with sliding windows, both in MNOS and HEE.
Seg	Multilayer Perceptron Neural Network algorithm for segment classification, both in MNOS and HEE.
N_i	MNOS inner node, $N \in \{Sw, Seg\}$.
F_N	Set of feature exctractors for a node $N \in T$.
I	2-dimensional RGB Image.
M_{GT}	Ground truth segmentation map for the image I.
D	Dataset.
$(I^{(k)}, M_{GT}^{(k)})$	Image and segmentation ground truth couple in the dataset D.
M_I	Etimated likelihood map: each pixel is assigned the probability it is foreground/object to segment.
M_I^N	Intermendiate likelihood map produced by a node $N \in T$.
h	Hypotesis for the estimation of a likelihood map M_I.
h^*	Intermediate hypotesis learnt by an inner node in a MNOS or HEE ensemble.
P_N	Meta-parameters of a node $N \in T$.

1 Introduction

1.1 Computer Vision and Object Segmentation

As humans, many tasks that rely on vision appear very simple to us. For example, it is very easy for us to count the number of animals in a picture, to perceive the depth of a scene or to be aware of the 3D structure of an object. Nevertheless we are very good at perceiving the different objects that compose a scene, separating them from the background and, eventually, change the focus of our attention in an instant and being able to separate the boundaries of a completely different object in the same scene. Many researchers tried and keep on studying the human visual system and how our brain is able to

perform such amazing tasks with such precision, but final answers are still far [1].

The reason why computer vision tasks are very complicated lies in the fact that computer vision is an inverse problem [2], as we are trying to find a unique solution to a task when not enough information is given. Then, it is necessary to resort to physics-based, computer graphics or probabilistic models in order to disambiguate between the possible solutions.

In computer vision, image segmentation is the task of finding groups of pixels that share some kind of common feature: it is a very challenging task whereas it also is a critical part in many applications, i.e. content based image retrieval. In statistics, this problem is known as cluster analysis and it is addressed by using formal models. Image segmentation is one of the oldest and the most widely studied problem: early techniques tend to use region splitting or merging approaches, which correspond to divisive and agglomerative clustering. This kind of solutions tend to focus on the inner characteristics of the pixel distributions but, if we want algorithms that mimic the human visual system, then something different is needed: a way to both include semantic, because we want to aswer questions likes: *where and what are the boundaries for a specific class of object given this set of images?*. A better solution is to optimize some global criterion that depend on the high level task the algorithm wants to solve.

Machine learning applied to computer vision gave a great contribution to tasks related to perception, such as object detection, recognition and segmentation. Statistical approaches distinguish between discriminative and generative methods: the former learn to model posterior probabilities in order to obtain a decision boundary while the latter focus on model likelihood and prior. Neural networks has also gained success in the last decade as they allow to empoy models that learn the intrinsic characteristics of the object of

interest directly from the reality, thus trying to address the segmentation problem with a biologically inspired approach. This solution has proved valid to face the many challenges that image segmentation implies, such as view point variations, illumination, occlusions, scale, deformation and background clutter without the need of defining specific solutions to handle such possible variations, which usually constitute the so called as intra-class variation that are very frequent in real world images.

In the same field, ensemble learning use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms: by using several algorithms at the same time and exploiting their interactions to solve a common task, it is possible to let several different machine learning algorithms collaborate and compensate their respective errors. In fact, even if two different algorithms has the same statistical performances, i.e. they obtain almost the same accuracy on a given task, it is reasonable to assume that they commit different prediction errors [3]. Systems that rely on multiple machine learning algorithms have the possibility to exploit the information produced by different classifiers by combining them in a common structure: one of the most famous ensemble in computer vision has been presented by Viola and Jones [4] for face detection. They built and ensemble of classifier using another famous model combination framework, called AdaBoost [5] which also introduced the concept of weak learners: algorithms that obtain poor performances alone but make possible to create a strong learner able to produce accurate results by combining them in an ensemble of learning algorithms. Many other recent solutions for image segmentation propose frameworks that combine algorithms that perform different tasks [6] or even the same algorithm multiple times [7] to solve a final and unique task achieving excellent results.

In this work a new ensemble of neural networks for object segmentation called Multi-Net for Object Segmentation is presented and a new model combination framework for combining any figure-ground segmentation algorithms, called High Entropy Ensembles, is derived. An extensive analysis of such framework is presented and a comparison with state-of-the-art model combination frameworks and segmentation algorithms is provided by using several standard datasets.

1.2 Summary

In Chapter 2 some ensemble method commonly applied in literature to figure-ground segmentation will be presented: bayesian classification and, especially, Bayesian Model Averaging (Sec. 2.1), boosting and Cascades of Boosed Ensembles (Sec. 2.2), Cascaded Classification Models (Sec. 2.3) and Auto-context (Sec. 2.4).

In Chapter 3 a new ensemble of neural network called Multi-net for Object Segmentation will be presented and discussed (sec. 3.1) by extending an already existing model devised for object detection (Sec. 3.1.2 and 3.1.3). The validity of this new algorithm is tested using a specific dataset constituted by images of clothes and a challenging standard dataset (Sec. 3.1.4). Finally, some real world application for this model are presented in Section 3.2.

In Chapter 4, the features used by Multi-net for Object Segmentation are briefly presented as well as how they are actually employed.

In Chapter 5 a new model combination framework is presented: starting from the Multi-net for Object Segmentation previously discussed, an algorithm to automatically configure the ensemble using greedy searches and an error backpropagation approach is presented (Sec. 5.1) followed by a subsequent solution that relies on randomness (Sec. 5.2): an analysis of the model (Sec. 5.2.3) and a comparison with standard datasets (Sec. 5.2.4) are provided.

Finally, conclusions are presented in Chapter 6.

2

Ensembles for object segmentation

Finding a single model that works well for solving a specific task often is not the optimal strategy. Usually, given a model M, several covariates M' of M are tried and the one that produces the best performance on the available data is kept and used. Once M' fits the data well and produce sensible predictions, it is used in generalization. The performance is often measured as a set of values, such as precision and recall, average precision or measures of accuracy that depends from the specific problem, such as overlap ratio for image segmentation or object detection.

The model that obtain the best compromise between precision and average deviation is finally employed but this strategy introduce a strong bias that comes from the choice of the model. Suppose to have a set of covariates of M that have similar performances. It is

reasonable to expect that they may commit different errors [3]: for some data cases the models that are not strictly the best may perform (even significantly) better.

In statistics and machine learning, ensemble methods are strategies to combine different learning algorithms in order to obtain a single model whose predictive performance outdoes the ones obtained by the single learning algorithms when used alone. This is a reasonable assumption as the choice of a specific model implies a bias, as stated above. In order to overcome this limit, a solution is thus to combine different models so that the resulting ensemble somehow "averages" the bias problem basing its own performance on the performance of its components.

An ensemble of a finite set of alternative models can assume different structures. As it is a combination of different models, an ensemble can be considered a graphical model where the topology of the structure describes the interactions between components. Usually, the topological structure of an ensemble is based on rejection rules or interactions among components.

Rejection rules, especially in classification, assume that besides a set of classes, a sample could belong to any classes but to a "no class" class. Thus, in a binary classification setting, like when we need to classify a data instance as "positive" or "negative", a solution based on rejection rule is usually employed as we can cast the "negative" class to a "no class" assignment. The main advantage given by these models is the computational efficiency. Ensembles based on interactions among their components, like a graphical model, use the results produced by each algorithm in a graph-like structure, where the output of each component is used by a subset of all the components in the ensemble. By using these kind of solutions, each components has access to much more information, thus potentially leading to very good performances, but also computational complexity becomes a

serious issue.

In computer vision, the idea of combining different segmentation algorithms together has already been successfully employed [8, 3, 9]. One of the most famous approach to build ensemble in the computer vision field is boosting [5, 10] that has been successfully employed to solve object detection and segmentation tasks [11, 12]. Moreover, it has a strong theoretical background [5].

Following boosting approaches, Cascades of Boosted Ensembles (CoBE) [13, 14, 4] are a good example of ensemble that focus on rejection. Here several classifiers, each slightly more complex than the previous, are used in a cascade in order progressively refine a result, thus saving a lot of computational resources by rejecting most of the candidates during the evaluation.

Another model combination framework recently published is Cascaded Classification Models (CCM) [6]: a combination of algorithms that perform different tasks are used to solve a segmentation problem, thus focusing on interactions between the components.

In the following sections AdaBoost, CCM and CoBE are briefly described along with Bayesian Model Averaging as they will be mentioned in the following chapters (especially in Chapter 5) because they are state-of-the-art model combination methods.

2.1 Bayesian model averaging

Bayesian Model Averaging (BMA) falls in the field fo bayesian learning. It is a technique that relies on bayes theorem and on the insight that the quantities of interest can be modelled by a distribution, thus it provides a probabilitic approach to inference. BMA can actually be considered a model averaging technique because it is a method to learn how to evaluate several hypothesis at once in order to produce a final classification result. In the most general case the hypothesis

are any probability estimators, in practical cases we can consider as hypothesis every classification algorithms that outputs a confidence score.

Given the training data D, bayes learning relies on bayes theorem:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \tag{2.1}$$

The probability calculated from Eq. 2.1 is called a posterior probability and it is the likelihood of the hypothesis h given the training data D. In a reasonable scenario, we have a set of different hypothesis H and we are likely to estimate the most probable $h \in H$, given the data D. According to [15], any such maximally probable hypothesis is called maximum a posteriori hypothesis (MAP) and it is defined by Eq. 2.2.

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D) \tag{2.2}$$

In this framework it is possible to calculate the most probable hypothesis given the training data, that is actually a function that can estimates either a probability score or a discrete value like a label, given the data D. Anyway, the question we want to acually answer is how to classify an unseen data istance $x \notin D$ in order to assign it a value $v_x \in V$, where it could be a label as well as a continous value like a probability. Instead of looking towards the prediction $h_{MAP}(x)$, we consider all the hypothesis $h_i \in H$ weighted by their posterior probability and for each possible value in V we want the most likely.

$$P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i)P(h_i|D) \tag{2.3}$$

$$v_x = \underset{v_j \in V}{\operatorname{argmax}} P(v_j|D) \tag{2.4}$$

Any system that actually classify a new instance according to Eq. 2.4 is called a *bayes optimal classifier*: it ensures that the probability the

new instance is correctly classified is maximized. The accuracy of the classification strictly depends from the hypothesis that are combined, so like every parametric model it is affected by a bias but the statistical background under this model combination algorithm ensures that there is no other classification method that can outperform a bayes optimal classifier, given the same data D and the same set of hypothesis H, along with the same prior knowledge.

A bayes optimal classifier cannot be practically implemented because of two main reasons:

- calculating the posterior probability for each hypothesis is infeasible as the hypothesis space is usually too large in practical applications: it is the major problem;
- from Eq. 2.3, calculating $P(h|D)$ (using bayes theorem, Eq. 2.1) implies calculating the unbiased estimate of the probability of the training set D given the hypothesis h, that is not trivial.

Also estimating the prior probabilities $P(h_i)$ is rarely feasible in practice, but it is possible to simplify the problem setting them all equally probable. So, given the previous issues, for real applications a suboptimal solution is necessary. Because the major problem is the time needed for the computation of the posterior probability from each $h_i \in H$, a solution can be to compute these quantities only for a subset $\bar{H} \subset H$.

A suboptimal solution is called BMA [16]. BMA approximates a bayesian optimal classifier by sampling hypothesis from the hypothesis space H and combining them using bayes theorem. The most restrictive version of this approach is known as Gibbs algorithm [17, 15] and it just draws a single hypothesis at random, according to the posterior probability distribution over H: given an instance to classify, it just applies this hypothesis to it. Surprisingly, it is proven that this

strategy, under certain conditions about the sampling strategy, has an expected misclassification error that is at most twice the one of the bayes optimal classifier [18][1]. Anyway, in BMA, the subset $\bar{H}$ are usually sampled using Monte Carlo techniques such as Markov Chain Monte Carlo [19]. This suboptimal approximation can be practically implemented and it is theoretically correct as it has a strong statistical background. In litterature there is some arguing about the possibility that BMA actually promotes serious overfitting [20, 21] compared to other ensemble techniques such as bagging [22].

2.2 AdaBoost and Cascades of Boosted Ensembles

Boosting [23] is a model combination technique developed to reduce the bias problem in supervised learning and it has had significant ramifications in machine learning and statistics since Schapire [10] proved its significance. Boosting techniques aim to create ensembles using a strategy similar to bagging [22], so with data resampling to train different parallel classifier, where the final decision rule is usually a majority voting. In boosting, the classifiers are trained one after the other and resampling for each of them is performed in order to provide the most informative data cases to each consecutive classifier. In the standard setting, boosting is applied to binary classification problems. Given a dataset D, each iteration of boosting creates three classifiers C_1, C_2 and C_3. The first classifier C_1 is trained using a random subset of the dataset. The subsequent clas-

[1] Actually this result is very interesting if we assume a prior distribution over the hypothesis space that is uniform because by applying just a random hypothesis drawn from the hypothesis space yields at most twice the expected error produced by a bayes optimal classifier.

sifier C_2 is trained on another subset of the dataset, where half of it is correctly classified by C_1 and the remaining is misclassified. This strategy actually provides C_2 a set of training samples that are very informative given the performance of the previously trained classifier. Finally the last classifier C_3 is trained using instances from the dataset on which both C_1 and C_2 disagree. As previously stated, the final decision rule for unseen instances is a majority voting.

Schapire also proves that this algorithms has an upper bound that is related to the error of the single classifiers ϵ. Precisely, he demonstrates that if the error of the algorithms is less than 0.5, than the error of the resulting ensemble is upper bounded by ϵ. So, in order to have an improvement we need algorithms that perform at least slightly better than random guessing: this is the concept of weak lerner [10], that is an algorithm that alone is not very good in a particular task. In contrast, the whole ensemble is called strong learner.

The problem of boosting techniques is that it is constrained to binary classificaiton problems. Adaptive Boosting (AdaBoost) generalize boosting to multi-class and regression problems [5]. It is probably the most known boosting technique as it is widely used for different applications in several fields [24, 4, 5, 25, 26, 27].

In AdaBoost for multi-class classification, the set of weak classifiers are trained using the bootstrap technique, that is each subsequent classifier is trained using a subset of the training data where the samples that were misclassified by the previous classifiers are more likely to be included.

The training subset for training the first classifier C_1 is sampled using a uniform distribution over D. C_1 is trained to produce the hypothesis h_1 using this subset and, once it has been trained, the error is computed by summing up the distribution weights of the

misclassified instances and it must be less than 0.5: it is proven that this constrain actually forces each subsequent classifier to correct at least one mistake made by the previous model. Next, this error (properly normalized) is used to update the distribution over D used to sample the training data in the bootstrap procedure. Once all the classifiers are trained, they constitute the ensemble and the final decision on a data instance is decided by majority voting, where the vote of each classifier C_i is weighted by its normalized error.

AdaBoost (and boosting in general), like BMA, has a strong theoretical background which ensures two important facts:

- the ensemble error monotonically decreases as new classifiers are added;
- it is surprising resistant to overfitting.

Both these considerations are not obvious as the former may appear to go against the conventional wisdom, because adding complexity to the model beyond a certain limit would lead to overfitting the data [28].

A Cascade of Boosted Ensembles (CoBE) [13, 14, 4] is a cascade of algorithms that actually are strong learners trained using a boosting technique. The most known application of a CoBE is with no doubt the Viola and Jones algorithm for face detection [4]. This work is a milestone for ensemble learning as it introduces a fast and reliable framework, along with other outbreaks such as the integral images for fast feature computation of Haar-like features, discussed in Section 4.3.

The general structure of a CoBE classifier is shown in Figure 2.1. Boosting is used to build an ensemble of classifier with increasing complexity, where each component learn a rejection rule. Like a focus-on-attention mechanism, the first classifiers learn to classify

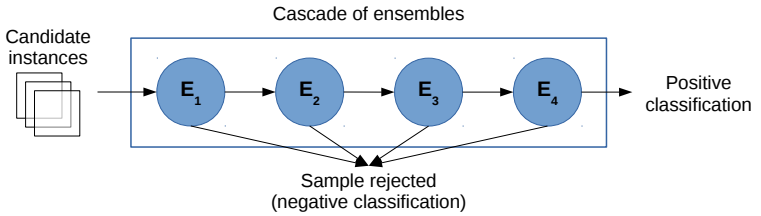


Figure 2.1: *Cascade of Boosted Ensemble structure.*

the instances achieving a very low false negatives rate: it allows to filter out a large number of samples in this early stage without loosing positive samples, thus speeding up the computation and reducing the number of instances to evaluate in the subequent classification steps.

This architecture gives some formal advantages regarding the false positive rates and the detection rate[2]. Because each classifier completely depends from the output of its ancestor, the false rate of the cascade is the product of the false rates of each classifier, and the same holds for the detection rate. So it is very easy to lower the false positive rate but, at the same time each classifier must be very good to achieve a good overall detection rate.

2.3 Cascaded classification models

CCM [6] is a framework for combining different algorithms that perform different tasks simoultaneously exploiting the interaction among them to improve their performance. The most interesting aspect of this method is that it allows to include algorithms into the framework as black boxes, without the need of understanding how they work or modifying them.

[2]By detection rate I mean the ratio of the positive classifications over the actual true positive in the dataset.

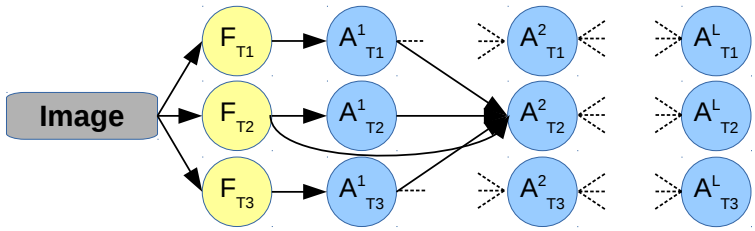


Figure 2.2: *Cascaded classification Model structure L-CCM. Features for each task $T_i \in T1, T2, T3$ are extracted and classifiers of the first tiers just rely on them. Classifiers on the next tiers rely on the classification results produced by all the classifiers in the previous tiers along with the features for its task.*

In order for an algorithm A to be used inside this framework, they must comply with a simple interface:

- it must be possible to train A on data (if necessary);
- new features can be added to A;
- provided data instances, it must produce a classification of this instance.

Once it is possible to treat each algorithm as a black box, the framework consists of an ensemble of algorithms which perform different tasks organized in tiers (see Figure 2.2): a CCM with L tiers is called L-CCM. First af all, features for each task are calculated. Then, the first tier is trained in isolation as it only rely on the features extracted from the input data. Next tiers are trained using the raw features extracted from the input data along with the output af all the classifiers at the previous tiers. Finally, in generalization, given an unseen data instance, classification is performed by predicting the most likely (MAP, see Section 2.1) assignment in the final tier.

This model, as will be discussed in Chapter 5, has a lot of similarities to the solution presented in this thesis. First of all, the idea of the algorithms as black boxes, providing an high level interface to comply with, that is a easy way to incorporate different algorithms in a framework which wants to exploit interactions between the components: this simplicity is a remarkable aspect. Second, this work proves that, when computational time is not an issue, interactions between the algorithms produces better results then using rejection rules, as all the tasks considered improved.

Anyway, such cascade is prone to overfitting if a lot of features are used or a lot of tiers are stacked. In Chapter 5 a similar problem will be discussed and solved using the model combination method proposed in this work.

2.4 Auto-Context

Auto-Context [7, 29] is not properly an ensemble of algorithms. In fact, it is a model to perform figure-ground segmentation originally developed for medical imaging that use stacked classificators in order to improve the segmentation performance at each stage. The algorithm targest the posterior distribution using a supervised approach: it tries to learn the marginals of the posterior which can be referred as segmentation maps: it actually learns to compute the likelihood that each pixel belongs to the foreground/object of interest given a training set, like what it is performed in a MAP setting (see Section 2.1). It solves the problem of computational efficiency by using stacked classifiers. The main breakthrough is that, at each stage, it relies both on the sub-window centered at a certain pixel and a large set of context pixels that are automatically selected by the learning procedure. Each classificator is able to integrate the image features from the sub-window with the raw values from the context pixels,

that are actually an estimation of the marginal distribution of the likelihood that a pixel belong to the foreground. It is proved that as a new classifier is learned, the algorithm converges to the marginal distribution of the pixels maintaining a manageable computational efficiency.

In other words, each classifier is trained to produce a object segmentation map where each pixel is assigned the likelihood that it belong to the object. The algorithm then manage to add classifiers that, given the input image along with a set of context-pixel from the previous classifier prediction, and it produces a new segmentation map that tend to improve in quality. The importance of this work is that it provides a theoretical proof that the learning converges to the objective function (the marginal distributions) in such a framework that stacks classifiers in order to increase the performance and monotonically decrease the training error.

3

Ensemble of Neural Networks for Object Segmentation: MNOS

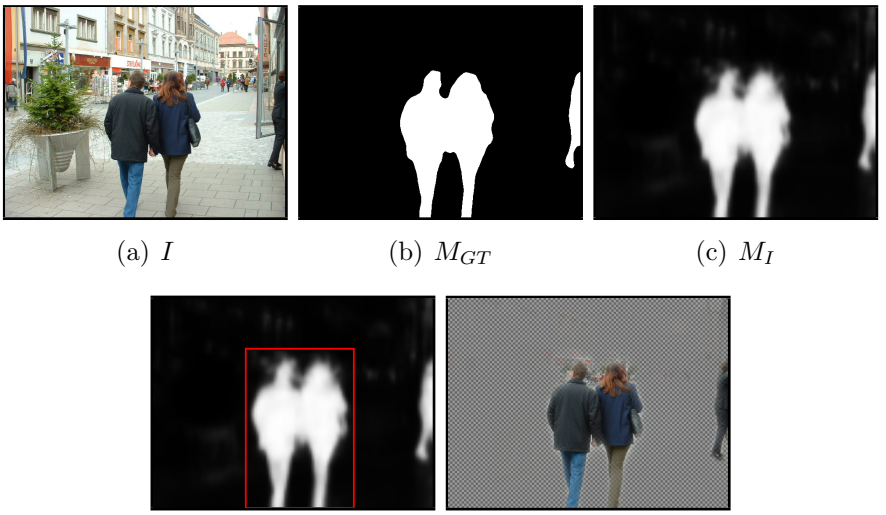
In this chapter, an ensemble of neural networks for figure-ground image segmentation is presented. As previously discussed in Chapter 2, model combination frameworks and ensemble of algorithms are widely used in computer vision and in image segmentation. Many works in literature prove that both cascades of algorithms, like Auto-context (Section 2.4) or CoBE (Section 2.2), and framework that focus on interactions between algorithms, like CCM (Section 2.3) help improving the quality of the results for several tasks. For many of them it is also formally proved.

Following this idea, this chapter will present an ensemble of neural networks initially designed for the object detection task called Multi-

net for Object Detection (MNOD). Initially, the problem a MNOD solves is formalized in Section 3.1.1. The idea behind MNOD, discussed in Section 3.1.2, is extended in order to perform figure-ground object segmentation: Multi-net for Object Segmentation (MNOS) is then presented in Section 3.1.3. The performance of the MNOS is discussed in detail in Section 3.1.4: MNOS is employed both on a standard dataset widely used in literature and on a specific dataset provided by the industry as it has been employed in real applications: in Section 3.2 some of them will be discussed.

3.1 From object detection to segmentation: MNOS

Multi-net for Object Detection [30] is an ensemble of neural networks that interact in a tree topology. This model is biologically inspired, like other works in literature related to machine vision [31, 32]: in our visual perception system, a neuron of the visual cortex receives a signal from the retina (a bottom-up low level stimolous) and a signal from an object model concept (top-down priming signal) that comes from prior knowledge and it activates depending from the strength of these two kinds of signals and from its activation threshold. The neurons are then organized in a hierarchical structure. Following the Gestalt theory, our visual system relies on prior knowledge that help identifying the objects of interests by focusing on salient features and inferring missing information. This "unconscious inference" process is the priming signal that along with the raw visual features allows the understanding of a scene. In the MNOD these two aspects of the recognition process are taken into account and integrated.



(a) I (b) M_{GT} (c) M_I

(d) Detection (e) Soft segmentation

Figure 3.1: *Given (a) the original image I and (b) its ground truth map M_{GT}, (c) a function $h(I) = M_I$ returns a pixel-wise likelihood estimate. M_I can be used both for (d) detection or (e) to obtain a soft segmentation map.*

3.1.1 Problem definition

A formal definition of the problem addressed by the MNOD follows. Object detection can be reduced to a classification problem: let I be a discrete color image and M_{GT} the ground truth map such that

$$\forall y_i \in M_{GT} : 0 \leq i < |M_{GT}| \quad \Rightarrow \quad M_{GT}(y_i) = \begin{cases} 0 & \text{if } y_i \text{ is background} \\ 1 & \text{if } y_i \text{ is object} \end{cases} \tag{3.1}$$

so that each y_i is a point in M_{GT}. We want to learn a function $h : I \mapsto M_I$ such that the following holds:

$$\forall y'_i \in M_I : 0 \leq i < |M_{GT}| \;\; \Rightarrow \;\; M_I(y'_i) \in [0, 1] \tag{3.2}$$

so that

$$E(I, M_{GT}) = \sum |h(I) - M_{GT}| \tag{3.3}$$

is as small as possible. $E(\cdot, \cdot)$ in Equation 3.3 is an error function that measure the distance between the two maps $h(I) = M_I$ and M_{GT}.

Figure 3.1 shows all the concepts just introduced. M_I is actually the likelihood that each pixel belong to the object to detect, that is the foreground in figure-ground segmentation. This map can be used both for detection, i.e. to infer the location of the object, or as a soft-segmentation map. Equation 3.3 actually measures how well M_I (Fig. 3.1(c)) approximates M_{GT} (Fig. 3.1(b)), so it actually is a measure of the quality of the hypothesis h.

3.1.2 The MNOD algorithm

A MNOD is a learning algorithm that learns the hypothesis h in a supervised setting. Instead of learning a unique h, it learns several different hypothesis h^* and then combine them in order to obtain a new hypothesis that perform better than the single ones. The MNOD is a an ensemble of neural networks T structured as a tree. Figure 3.2 shows an example of MNOD: the nodes are computational units and the directed edges represent the interactions between these units. Two kind of nodes form this ensemble, *leaf nodes* and *inner nodes*, which have different roles:

- *Leaf nodes* are feature extractors: given the input image I each leaf extracts a specific feature patterns; each feature extractor

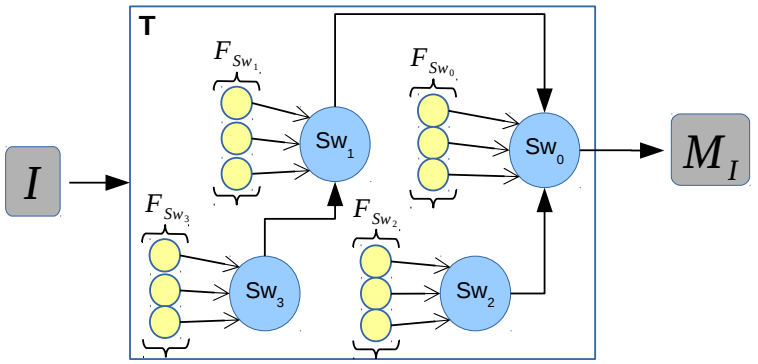


Figure 3.2: *MNOD Structure. Each inner node Sw_i takes in input the feature patterns calculated by its feature extractors F_{Sw_i}, the original image and eventually the output of another inner node, and returns an estimation of the likelihood that each pixel in the image belongs to the foreground.*

f_j is child of a specific *inner node* Sw_i and $f_j \in F_{Sw_i}$; it must also be possible to represent the feature patterns as an image in the same space of I.

- *Inner nodes* Sw are classifiers that learn a function $h^* : I \mapsto M_I^{Sw}$.

Feature extractors calculate feature patterns directly from the input image I and the features that have been used are discussed in Chapter 4. Following the biological perspective given in Section 3.1, they provide the bottom-up low level stimolous to the *inner nodes*. The *inner nodes* actually are Multi-layer Perceptron (MLP) Neural Network [33] and learn to reproduce a function h^*; a neural network Sw_i receive as input a set of features patterns by all nodes in F_{Sw_i} calculated from I and, eventually, one or more maps $M_I^{Sw_j}$, where Sw_j is a child node of Sw_i in T: all the input data is processed by

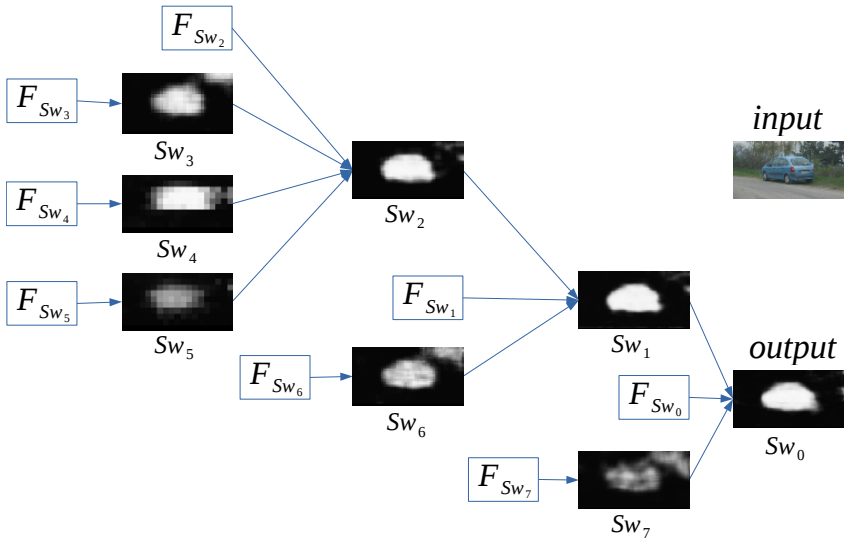


Figure 3.3: *Each node Sw_i receives in input the feature patterns calculated by the nodes in F_{Sw_i} and eventually one or more likelihood maps by its child nodes. It returns its hypothesis. Each node processes the images at a different scale w and with different receptive field size s and the figure shows the improvement of the likelihood map as the computation go from the leaf nodes to the root Sw_0. Finally, the map computed by Sw_0 is the output likelihood map M_I.*

a standard sliding window approach, which constitutes the receptive field of the neural network. For each dense positioning of the sliding window, the patterns for a node Sw_i are created by juxtaposing the values that lies under the sliding windows in all the input likelihood maps and feature patters. The neural network is trained to produce an estimate for all the pixels that fall in the sliding window receptive field, so the output likelihood map $M_I^{Sw_i}$ is obtained by averaging all the prediction for each pixel in I. These Neural Networks provide the top-down primal signal to the ensemble as they hold the knowledge

to estimate the pixel-wise likelihood discussed in Section 3.1.1.

A MNOD tree T needs a training procedure as it uses supervised learning algorithms. Let D be a training dataset such that

$$(I^{(k)}, M_{GT}^{(k)}) \in D, \quad 1 \leq k < |D| \tag{3.4}$$

where $I^{(k)}$ is a training instance and $M_{GT}^{(k)}$ is the target likelihood map for each k. Each $Sw_i \in T$ is recursively trained with the data D: starting from the root of T, each node starts by training its child nodes; *leaf nodes* do not need to be trained. By doing so, the first nodes Sw that are actually trained are the ones that have no child nodes except the ones in F_{Sw}. For example, the nodes in Figure. 3.2 are trained in the following order: $Sw_3, Sw_1.Sw_2, Sw_0$. In generalization, a similar recursive procedure is performed: given I, the root node is asked to predict $h(I)$. Starting from its children, each node in T is recursively asked for its hypothesis $h^*(I)$ and each *leaf node* computes their features. Once the intermediate hypotheses h^* have been calculated, the output of the root node $h(I) = M_I$ is computed. Finally, a threshold is applied to M_I in order to get a binary map and the bounding boxes are determined around the regions obtained by applying a threshold (like Fig. 3.1(d)).

The architecture for a MNOD is quite simple but there are a lot of meta-parameters as the algorithms involved include several feature extraction algorithms and MLP neural networks. There is a parameter that must be defined for each node, regardless its nature, that is the image size w as the input of a node, both likelihood maps and features, are resampled to the same size w before applying the computation, maintaining the image proportions (the major dimension is resized to w and the minor one is resized proportionally).

The MLP neural networks are trained using the Resilient Backpropagation (RPROP) [34] algorithm, that is much faster than standard backpropagation: as there can be a large number of nodes to

train, RPROP provides a good speedup while maintaining good performances [35]. Since the nodes Sw are trained to learn a function h^* and to minimize Equation 3.3, RPROP is used with the mean square error function. The meta-parameters P_{Sw_i} for each *inner node* $Sw_i \in T$ are the ones for the RPROP, the sliding window size s and w.

The last component are the *leaf nodes*: their parameters P_{f_i} depend on the algorithms they implement for calculating the feature patterns (see Chapter 4).

3.1.3 A new node for object segmentation: MNOS

The likelihood map M_I produced by a MNOD ensemble is good for object detection, that is locating the area of the image where a particular object is present. It is similar to saliency detection, even if it does not work exactly like an algorithm for saliency estimation [32], because it does not assign high probabilities to image locations that should attract the attention of the observer but it actually performs a pixel-wise classification of the image given a class of object which the algorithm had been trained to search for. The likelihood map M_I allows a good detection [30] as it usually predicts at least the coarse location of the object.

The MNOD can also be used for object segmentation: Figure 3.1(e) shows that by using the likelihood map to generate a soft segmentation map leads to a fuzzy object segmentation. The likelihood map in the example is a case of very good estimate because it almost entirely cover the objects, but there are two issues:

- most of the times, the "clouds" in M_I are very coarse as, like saliency maps, it is just useful to identify the object location;

- even if M_I is sufficiently accurate, it leads to a very fuzzy segmentation.

The first issue is due to the sliding window approach used with the MLP neural networks: it produces a resilient detection but a very inaccurate estimation of the object details. Then, the second issue follows as it is rather impossible to obtain a likelihood map where the regions have crisp borders because of the nature of M_I.

In order to overcome the limitations of the MNOD for segmentation tasks a new solution must be devised: as the main problem lies in the fuzziness of the likelihood maps M_I when used as segmentation masks, the nodes which use the sliding window approach for the neural networks are no longer suitable. Following the idea of [36], instead of learning and predicting the likelihood that each pixel belongs to the object, a new node *Seg* that relies on the classification of image regions, called segments, is used. In [36] several rough segmentation hypothesis are ranked and used to obtain a final good object segmentation: it is very computational demanding as computing thousands of hypothesis, even by using the maxflow-mincut based algorithm CPMC, is infeasable if we want to use it in an ensemble.

The solution proposed [37] calculates a partition of the image instead of looking for several hypothesis and learns to predict the likelihood that each region of the image belongs to the object.

Figure 3.4 is an overview of a MNOD node *Seg*; it has the same role as a *inner node Sw* described in Section 3.1.2 as it complies to the same interface: it takes in input the image I, the features from F_{Seg} and the output maps produced by the other child nodes; it returns its own pixel-wise likelihood estimate. The image I is partitioned into a set of segments $S = \{s_1, \ldots, s_N\}$ such that $\forall i \neq j$ with $1 \leq i, j \leq N$, then $S_i \cup S_j = \varnothing$ and $\cup_{i=1}^{N} s_i = I$. This partition S is calculated by using the k-means clustering algorithm [38] where each point in I is

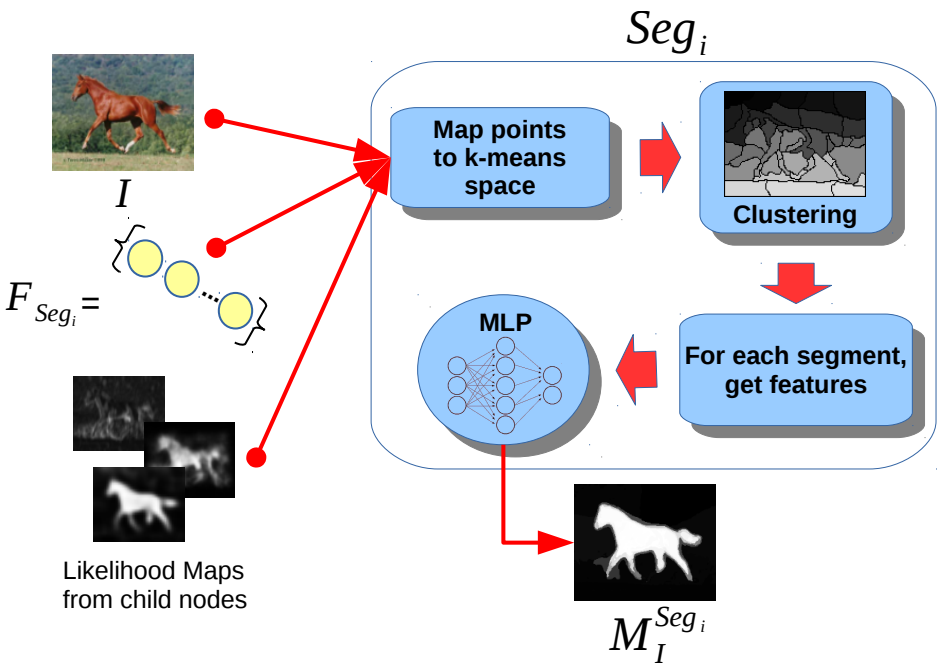


Figure 3.4: *A node Seg_i takes as input the image I and the output of its child nodes (both feature patterns from F_{Seg_i} and likelihood maps). It calculates the segments by clustering the image points and then a MLP neural network is used to assign them a likelihood estimate.*

mapped into a new space whose dimensions depend on the number of children of Seg_i (both the features F_{Seg_i} and other *inner nodes*): three values are about the three color channels of I, two values about the (x, y) position on the image canvas, the other are the intensity values resulting from each child node, feature pattern or likelihood map (as stated in Sec. 3.1.2, both likelihood maps and feature patterns are considered like images).

Each segment must be assigned the likelihood it belongs to the

object like what the nodes Sw do but, instead of generating the patterns for the classifier using the sliding window approach, several features about the geometry and intensity values distributions are used. Precisely:

- area and perimeter of the region, the ratio of the perimeter over the area;
- center of the bounding box of the segment;
- hu moments [39];
- quantized histogram of the intensity values of the segment;
- quantized histogram of the intensity values of the image area surrounding the segment.

Most of them are also used by [36]. This set of features is used by a MLP neural network to estimate the likelihood the segment belongs to the object.

The output of a node Seg_i is a likelihood map M_I: as the set of segments forms a partition of I, all the points in I are assigned the likelihood of the segment they belong to.

A node Seg_i actually implements a function $h^* : I \mapsto M_I$, where M_I satisfies Equation 3.2, and it can be used as *inner node* in a MNOD ensemble obtaining a MNOS [37].

3.1.4 Analysis of the MNOS

The two different nodes Sw and Seg learn the same kind of function h^* but they rely on different kind of information to calculate their hypothesis. Sw nodes uses the sliding window strategy to compose the data for the MLP neural network and estimate the likelihood in a pixel-wise manner: the sliding window approach allows to take into

(a) I

(b) Segmentation using the likelihood map by a node Sw

(c) Segmentation using the likelihood map by a node Seg

Figure 3.5: *Segmentations by the likelihood maps produced by the two nodes Sw and Seg: (a) the original image I, (b) segmentations using the likelihood map of Sw (left) obtaining a soft segmentation map (center) or by thresholding it by half the range (right), (c) the same but using the likelihood map of Seg.*

account the context of each pixel inside the receptive field (which size is $s \times s$) but actually the predictions are pixel-wise as the neural network for each sliding window has to give an estimate for each point in the receptive field. On the other hand, *Seg* nodes compute a single likelihood estimation for an entire area actually performing

Figure 3.6: *Sample images from the Drezzy Segmentation Dataset.*

the classification of an entire region of the image at once, so that the borders between different regions produce crisp segmentation masks instead of "clouds" of points.

This is shown in Figure 3.5. Given the image I, we want to separate the foreground, that is the object of interest, from the background. The two likelihood maps are obtained respectively using a node Sw (Fig. 3.5(b)) and a node Seg (Fig. 3.5(c)); the segmentations are obtained by applying the likelihood maps to I with and without a threshold (set to half the range of the likelihood maps values), in order to obtain a soft and an hard segmentation respectively. From Figure 3.5 it is easy to notice the different characteristics of the two nodes: as stated before, Sw produces a fuzzy segmentation mask that is not suitable for obtaining a sharp foreground-background object separation, even by applying an hard threshold; the node Seg, as it has been devised to solve this issue, produces sharp and precise boundaries around the different regions and allows for a better separation.

In order to build a MNOS ensemble using the two nodes described so far, it is necessary to study how they work and their behaviour when they interact together. The following experiments were performed using the Drezzy Segmentation Dataset [40], which consists in 8 classes of clothing images from commercial products sold in the web, with about 200 images per class, 200×200 pixels. Some sample images are shown in Figure 3.6. This is a dataset built from the

database of the company that owns the products which the MNOS has been applied to (see Sec. 3.2): the object is always in the center of the image and the background varies from uniform colors to people wearing the clothes, logos and mixed and irregular colors.

Finally, the MNOS ensembles are tested using the figure-ground segmentation variant of the VOC dataset [41]. As in other figure-ground segmentation works [42, 43, 44], we fuse the ground-truth maps for the 20 object classes into single segmentation maps. For both the dataset, the quality measure employed is the pixel-wise Intersection over Union (IoU) between the ground truth mask M_GT and the segmentation map produced by applying a binary threshold $thresh(\cdot)$ to M_I (like in Fig. 3.5). Given and image I, its ground truth mask M_{GT}, a MNOS T that produces a likelihood map M_I, then

$$IoU_{T,I} = \frac{|M_{GT} \cap thresh(M_I)|}{|M_{GT} \cup thresh(M_I)|} \tag{3.5}$$

where $|\cdot|$ denotes the area, which is the number of pixels in the region.

Table 3.1 and Figure 3.7 shows the results obtained on the Drezzy Segmentation Dataset. For these experiments, MNOS ensembles with 4 layers (tree of height 4) are built for each class and two different strategies are employed:

- **MNOS only Seg**, only composed by nodes Seg;
- **MNOS Sw + Seg**, where the first two layers are composed by nodes Sw and the last two by nodes Seg.

The first solution produces good results on simple images, where the separation between the object and the background is clear but bad results where it is not: complex backgrounds, the presence of a person that wears the cloth or high contrasts. This issue is due to the

Table 3.1: *IoU results on the Drezzy Segmentation Dataset using two configurations: a MNOS with only Seg nodes, a hybrid ensemble with Sw nodes in the first layers and Seg nodes in the last layers. Last column uses the segmentation masks from the latter model to initialize the GrabCut.*

	IoU		
Class	only *Seg*	*Sw* + *Seg*	GrabCut
Bags	0.79	0.79	**0.89**
Shoes	0.78	0.88	**0.93**
Hats	0.64	0.63	**0.82**
Ties	0.78	0.82	**0.92**
Man Clothing	0.68	0.73	**0.82**
Man Underwear	0.39	0.65	**0.80**
Woman Clothing	0.58	0.63	**0.67**
Woman Underwear	0.35	**0.67**	0.61
Average	0.62	0.72	**0.81**
Std. Deviation	0.17	0.09	0.12

fact that *Seg* nodes are unable to catch features at a lower granularity than general characteristics of homogeneous regions. The second strategy is devised to overcome this limitation as it uses *Sw* nodes in the first layers: processing the information at pixel level by using the sliding window followed by a classification of segments using the *Seg* nodes, which rely on features extracted from entire regions of the image, is an implicit aggregation process while the information flows through the tree from the leaves to the root, in a bottom-up process. Thus, the root node is a *Seg* node, so the final likelihood map used for the segmentation maintains the properties discussed in

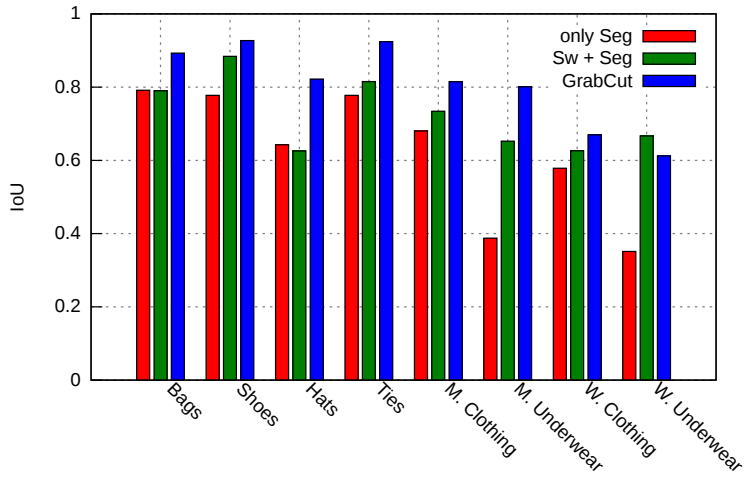


Figure 3.7: *Results on the Drezzy Segmentation Dataset*

Section 3.1.3. Table 3.1 shows that this approach solves the problems occurred when using the first strategy, actually obtaining good segmentation results on the problematic classes while preserving the quality on the others, which allows for an higher average segmentation accuracy and a better reliability, as the standard deviation is lower. A MNOS is very fast in generalization as it only needs in average less than 100 ms per image from the Drezzy Segmentation Dataset. The training time depends from the parameters of each neural network that composes the ensemble: training a 4-layer MNOS like the ones used in the previous experiments takes from 2 to 3 hours on an average Desktop PC.

An interesting way of using the MNOS is to employ the likelihood map to initialize the GrabCut algorithm [45]: it is an interactive segmentation algorithm that has proved to be very effective once it is well initialized by the user. The GrabCut need examples of background and foreground areas in order to calculate statistics

Table 3.2: *Results on the VOC2010 Dataset, figure-ground variant, using an ensemble with Sw nodes in the first layers and Seg nodes in the last layers. Refinement with GrabCut is also used.*

Class	MNOS	GrabCut	**Class**	MNOS	GrabCut
Airplane	0.36	**0.56**	Dining Table	**0.26**	**0.26**
Bicycle	**0.15**	0.13	Dog	0.35	**0.38**
Bird	0.24	**0.37**	Horse	0.31	**0.42**
Boat	0.31	**0.36**	Motorbike	**0.51**	0.49
Bottle	**0.21**	0.19	Person	0.33	**0.36**
Bus	0.51	**0.57**	Potted plant	0.19	**0.24**
Car	0.37	**0.38**	Sheep	0.31	**0.32**
Cat	0.37	**0.39**	Sofa	0.26	**0.28**
Chair	0.05	**0.10**	Train	0.50	**0.52**
Cow	0.41	**0.54**	TV monitor	0.18	**0.27**

Average IoU	
Küttel and Ferrari [42]	0.48
Carreira and Sminchisescu [43]	0.34
Rosenfeld and Weinshall [44]	0.46
MNOS Average	0.31
MNOS + GrabCut Average	0.36

and, iteratively, it refines the segmentation using the graph cut algorithm. The likelihoods of each pixel calculated by the MNOS can be used to inizialize the Gaussian Mixture Model (GMM) [46] used by the GrabCut: available implementations [47] allows for four degrees of likelihood: (i) background (ii) likely background (iii) likely foreground and (iv) foreground. It is possible to map the likelihood

values to a quantized space that matches this labels and initialize the GrabCut accordingly. Table 3.1 and Figure 3.7 shows also the results obtained by applying this kind of refinement step: it turns out to produce a sensible improvement in all the classes but the last one, that is also the more complex and the one the MNOS obtain the worse results. The reason why this happens is that the GrabCut performs well if the initialization is good but it can worsen the results if it is not, because it learns bad statistics about the background and the foreground color distributions. This fact makes this solution a little bit more unstable than just the MNOS segmentation approach but, in average, produces far better results. Another price to pay using the GrabCut is the computational time: 4 iterations have been used to obtain the results presented with an average time of about 500 ms per image.

Table 3.2 presents the results obtained on the VOC2010 [41] dataset. The results are compared with other works that also use the figure-ground variant on the VOC2010 dataset [42, 43, 44] for segmentation. The same considerations made for the Drezzy Segmentation dataset are still valid: while on the previous simple dataset the performances are good, it is clear that in presence of complex scenes the MNOS fails to obtain sensible results. In Chapter 5 the main reason for these poor performances will be discussed and a solution to make MNOS competitive with state-of-the-art solutions will be presented. Moreover, Table 3.2 will be expanded by Table 5.2 in order to allow a direct comparison with the solution presented in Section 5.1.

3.2 Applications

The MNOS has been devised in order to solve the segmentation problem for images of cloths sold on the web. Specifically, it has been em-

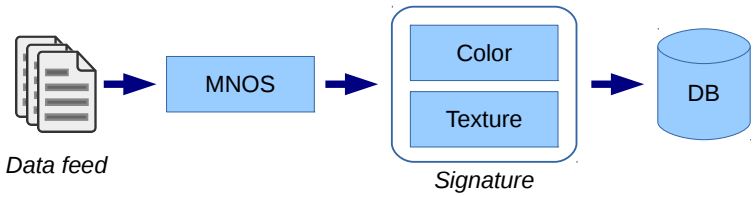


Figure 3.8: *Drezzy indexing process: MNOS is used to segment the images of the products; on the segmented area a signature is computed, which is stored on the database for CBIR.*

ployed in the indexing process of the offers for a commercial website called *Drezzy*[1] [48].

Drezzy is a price comparison website that every day processes about 400.000 offers provided by hundreds of data feeds from different merchants. The website provides the functionalities to search for products and compare the different offers available on the web, also allowing the navigation by visual features such color and texture both by facets and by example: it is possible to filter the results by color or texture using the facets and, given a specific offer, to look for similar products. All these functionalities require information about color and texture that are extracted from the image of the product; the innovative aspect of the *Drezzy* indexing engine is that the extraction of color and texture information is fully automatic, with no manual tagging required, by using an approach similar to [49].

Figure 3.8 represents the indexing pipeline: given a data feed with the information about all the offers, the goal is to preprocess the data in order to prepare a signature for each offer that will be used for Content Based Image Retrieval (CBIR) tasks. For each offer, the image is segmented using a MNOS ensemble in order to isolate the area of the image representing the product and avoiding computations on

[1] `www.drezzy.it`

background pixels. This is a vital step as the signature of the object is computed on the area detected by the MNOS and it is composed by both information about color and texture. Color information is represented by two histograms of two spaces of representative colors derived from the ISCC/NBS system as proposed by the Inter-Society Council[2]. The first space is composed by 27 colors: this histogram is used to perform faceted queries, the second space is composed by 267 colors and, because it is more descriptive, it is used for similarity queries (queries by example) [50]. Texture information is cast to a classification problem: each image is convolved using a bank of 6 filters from which a 11-dimensional feature pattern is extracted. The classification is performed by a SVM [51] with gaussian kernel and it is twofold: fistly, the segmented image is classified; secondly, a set of patches is extracted from the segmentation area and are classified in order to build an histogram of the different labels assigned by the SVM.

Color and texture information form the signature af an image from a certain offer. Both are stored in a Apache Solr[3] database in textual form exploiting the features of Solr on textual contents to obtain an easy and effective retrieval system, as described in [50].

The previous functionalities for CBIR have also been exploited for an Android mobile application [52]. An API has been exposed in order to allow this application to perform queries by examples on the *Drezzy* database and browse the available offers on mobile devices. It allows to take a picture and instantly query for similar product by constraining the search on a specific category of clothes.

The last commercial application of MNOS is for gas meter detection

[2] `www.iscc.org`

[3] `http://lucene.apache.org/solr`

and digit segmentation. As described in [53], given a picture of a gas meter, like the ones of people's homes, a first MNOD ensemble is used to perform the detection of the counter that measures the amount of gas consumed. Once the counter has been located, the digits inside are detected and finally an MNOS ensemble is used to segment them from the background. This system has been employed by a business company for a project involving the automation of the gas meter counters acquisition.

4

Feature patterns extraction

In Chapter 3 an ensemble for Object segmentation has been discussed. It is a tree structure where the components are the *inner nodes*, which are classificators, and *leaf nodes*, which are feature extractors. The latter are algorithms for calculating feature patterns from the input image I for the former, in order to create the set of patterns for the classifiers.

In the following section the feature employed in an MNOS ensemble are briefly presented. As previously stated, it must be possible to represent the feature patterns as images; let f be a feature extraction algorithm such that $f : I \mapsto \vec{p}$, where $\vec{p}$ is a feature pattern. It must be possible to define a new function $g : \vec{p} \mapsto I_f$ such that maps the feature pattern to a single channel image space with the same size of I. The reason is that both the nodes Sw and Seg described in Sec-

tion 3 create the patterns for their neural networks from the image space: the former by positioning a receptive field on the image space and the latter by calculating features from region in the image space too: this solution allows to extract the information for the neural networks both from the likelihood maps M_I and the extracted features I_f using the same general strategy. One major constraint is also the computational time needed to calculate $g(f(I))$: each *inner node* uses a neural network that is very fast in generalization; they also have several feature extractors that would then have an extremely negative effect on computational efficiency if they are slow to compute the features. So complex and elaborated feature extraction algorithms are too slow, such has textons based approaches [54, 55, 56] or feature learning algorithms [57, 58, 59] which usually need a training procedure, so they have not been considered. On the other hand, simpler solution are preferred, such as different channels representations (i.e. CIE-Lab [60]), convolutional filters, wavelet and histograms are preferred as they are very fast to compute which are used by other works in literature for the same reason [61].

In the following sections many feature extraction algorithms will be briefly discussed as they are employed as *leaf nodes* by both the MNOS discussed in Section 3.1.3 and the High Entropy Ensembles (HEE) framework described in Chapter 5.

4.1 Color channels

One of the simplest feature that is possible to extract are the channels of different image spaces: the resulting feature is a single channel from a color space. We extract the single channels from RGB, CIE-Lab [60] and HSB color spaces, resulting in nine possible features extractors: three channels from each of these spaces.

4.2 Frequency filters

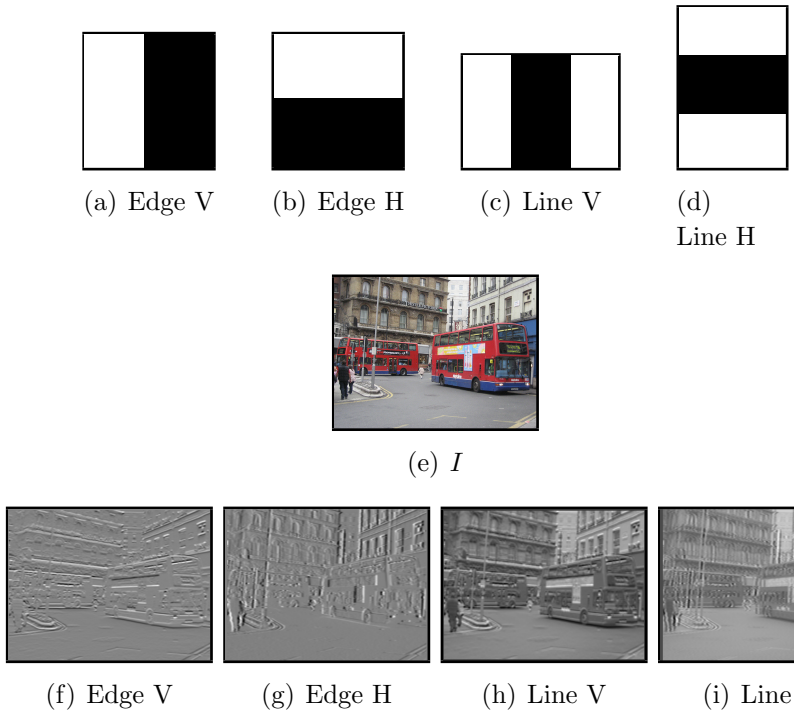
In image processing, frequency filters are operators that performs a cut of certain frequencies that compose the image signal. Usually there are two types of frequency filters: high-pass filters and low-pass filters [62]. The former remove the low frequency components from the image signal while the latter remove the high frequencies. So, the application of a low-pass filter produces a smoothed image as the components with high frequencies, that are high contrast regions, are cut off. On the other hand high-pass filters generates an image with only the high frequency regions.

Such filters can be obtained by applying the Fourier transform and then removing the components that are below (or above) the cutoff level. In order to create a feature extractor MNOS uses algorithms that allow to get the same results without the need of using the Fourier transform but using simple convolutions. As low-pass filters, a simple gaussian blur is used: kernel for the convolution is calculated by discrete sampling the values from a bidimensional gaussian distribution with the same spread σ for both the dimensions; for the latter Laplacian of Gaussian (LoG) is used, which means firstly to compute a gaussian blur and then a laplacian of the result, which is a kind of second order derivative.

The parameters needed for both this feature extractors are kernel size and the spread σ of the gaussian.

4.3 Haar-like features

Haar-like features were first introduced by [63, 24] for face detection using an ensemble of neural networks. The main peculiarity of this kind of features is that they are very fast to compute once the integral image is generated. As the name suggest, an integral image for a



(a) Edge V (b) Edge H (c) Line V (d) Line H

(e) I

(f) Edge V (g) Edge H (h) Line V (i) Line H

Figure 4.1: *Haar-like filters used by MNOS and HEE*

gray-scale image I is a matrix I_{int} with the same size of I where for each point $p = (x, y)$, $I_{int}(p)$ is the sum of all the intensity values in the rectangle area starting from $(0, 0)$ to (x, y). An haar-like filter for the MNOS are sliding windows like the ones in Figure 4.1: for each positioning on I, the area under the black part of the filter is subtracted from the area under the white part. So, in this setting f is the aforementioned procedure and generates a set of values, one for each positioning. This feature pattern can be mapped to the image space by defining a function g such that, by performing a dense sampling of I, it assign to each point in the image space the value obtained by computing the subtraction between the areas of the two

parts of the filters. This operation can be performed in costant time given I_{int}.

In the MNOS, four kinds of Haar-like features are used and they are the ones showed in Figure 4.1: two edge features (Fig. 4.1(b) and 4.1(a)) and two line features (Fig. 4.1(d) and 4.1(c)), for a total of four feature extractors. Figure 4.1 also shows an example for every type of haar-like feature used.

The only parameter for these feature extractors is the size of the stripes in the filters. For efficiency purposes, the integral image I_{int} is calculated once for each image if at least one haar-like leaf node is present in the ensemble T.

4.4 Edges detectors

Many feature extractors used by both the MNOS and HEE are edge detection algorithms. The simplest edge detection algorithms used are simple derivatives of the image I for x and y dimensions alone or both. They are actually implemented as convolutions with simple derivative kernels k such that $g(f(I)) = I * k$, where $*$ denotes a bidimensional convolution. The edge detection algorithms used are the following:

- x and y derivatives by respectively using $k = [-1; 2; -1]$ (column vector) and $k = [-1, 2, -1]$ (row vector);
- 45° and 135° edges by respectively using

$$k = \begin{vmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{vmatrix} \text{ and } k = \begin{vmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{vmatrix}$$

- x and y Derivatives of Gaussian (DoG), which consist on convolving I with a gaussian filter (as described in 4.2) and then performing onother convolution to obtain x or y derivative;
- Sobel filter;
- Canny Edge Detector [64].

4.5 Histogram of Oriented Gradients

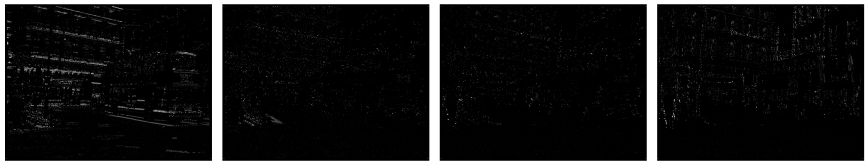
Histogram of Oriented Gradients (HOG) was first introduced by [65] for person detection. The algorithm, given a region of interest of I, creates an histogram of the unsigned gradient vectors magnitude by performing a quantization of the angles: in an HOG with i.e. 9 bins, each bins is the quantization by 20 degrees of the gradient vectors magnitude because $180/9 = 20$.

In order to match the contraint to use this feature for the MNOS and HEE, it is necessary to map this descriptor to the image space I by defining a function g, as described above. Given the image I and the number of desired bins n, this is done by the following algorithm:

1. The derivatives D_x and D_y of I are computed for the two dimensions as bidimensional matrices with the same size of I.
2. The angle step used for the quantization is computed as $\alpha_{step} = 180/n$.
3. n bin-matrices B_I^i, $0 \leq i < n$ are initialized with the same size of I, one for each bin, such that $B_I^i(x, y) = 0, \forall (x, y) \in B_I^i$.
4. For each point $(x, y) \in I$, the gradient magnitude and angle are computed respectively as

$$m_{x,y} = \sqrt{D_x(x, y)^2 + D_y(x, y)^2}$$

(a) I



(b) B_I^0 $0 \leq \alpha < 22.5$ (c) B_I^1 $22.5 \leq \alpha < 45$ (d) B_I^2 $45 \leq \alpha < 67.5$ (e) B_I^3 $67.5 \leq \alpha < 90$

(f) B_I^4 $90 \leq \alpha < 112.5$ (g) B_I^5 $112.5 \leq \alpha < 135$ (h) B_I^6 $135 \leq \alpha < 157.5$ (i) B_I^7 $157.5 \leq \alpha < 180$

Figure 4.2: *Hog features with 8 bins used by MNOS and HEE: all the results of the possible feature extractors are showed. For each image, the range of the gradient angle quantization are noted.*

$$\alpha_{x,y} = \arctan \frac{D_y(x,y)}{D_x(x,y)} \cdot \frac{180}{\pi} + 90$$

so that $\alpha_{x,y}$ and is expressed in degrees rather than in radiants and $0 \leq \alpha_{x,y} < 180$.

5. For each point $(x, y) \in I$, $i = \lfloor \alpha_{x,y} / \alpha_{step} \rfloor$ and $B_I^i(x, y) = m_{x,y}$.

The parameters for the HOG feature extractor for the MNOS and HEE are the number of bins n for the quantization and a bin index

i and, given an image I as input, it returns B_I^i as gray-scale image. Like what it is done with haar-like feature extractors, the set of bin-matrices are computed once if there is at least one HOG feature extractor node in the ensemble and each node share this set, thus resulting in very fast computations.

5

High Entropy Ensembles

In Chapter 3 an ensemble of neural network for object and figure-ground segmentation has been presented and discussed. It combines two kinds of nodes that learn to perform a classification task that is used to solve a segmentation problem by using two different strategies. Each node solves the problem exploiting the interactions with other nodes and the tree topology make the information flow from the leaves to the root node, whose output is the final segmentation hypothesis, in a sort of refinement process. That solution has been motivated by the industry as it has was devised for the indexing process of a website that exploits computer vision to assist the users during the navigation.

As discussed in Section 3.1.4, MNOS is good for the task it has been devised for but its performaces on a standard and challenging

dataset are not acceptable. The main problem is that the MNOS architecture needs a lot a meta-parameters and also the topology of the tree is an aspect to care for: the previous experiments used MNOS structures tuned by hand because it is impossible to perform both an optimal search or a grid search in such a huge parameter space, also considering the time needed to train a MNOS ensemble on a large dataset.

In the next sections both the problems of selecting the topology and the meta-parameters are addressed. In Section 5.1 an incremental algorithm for configuring a MNOS is presented: it is based on the idea of backpropagating the errors commited by a node and to add additional nodes to help correct them. This attempt to find a solution to the aforementioned problem needs a formal formulation, presented in Section 5.2.1, which introduces a more general framework than the one represented by the MNOS. In Section 5.2.2 an algorithm that finds a configuration for an ensemble of algorithms called HEE is presented. Finally Section 5.2.3 and 5.2.4 discuss why the algorithm proposed is so effective and its effectiveness is proved using several standard and challenging datasets.

5.1 BPConf: model combination algorithm through error backpropagation

In literature, several strategies have been proposed for searching through huge parameter spaces, such has genetic algorithms [66] or ant colony optimization [67]. Both this approaches are meta-heuristics used in optimization in order to find an approximate solution in a parameter space that is too wide for greedy searches or too complicated for analytical approaches. On the other hand, in model combination frameworks several algorithms have great popularity in literature, such as boosting [5, 10] and CoBE [13, 14, 4], discussed

in Chapter 2, but all of them do not address this problems as the structure of the ensemle is fixed as they are not graphical models.

The MNOS needs a completely different algorithm because different issues arise: the topology of the model must be defined, that is how actually the components of the tree interact. Moreover, there are meta-parameters to tune, such has the scale w for each node, the receptive field size s for Sw nodes and the feature extractors. Both this aspects need to be addressed by a general strategy.

A first solution to address both these problems is an algorithm called *BPConf* [68]. It uses an incremental approach to build the MNOS tree while performing a greedy search in the parameter space for each component it adds. *BPConf* needs a dataset D such as the one described by Equation 3.4, from which a validation set D_{val} and a training set D_{train} such that $D_{val} \cup D_{train} = D$ and $D_{val} \cap D_{train} = \varnothing$ are extracted. It also needs a set of constraints that drive the optimization process:

- the maximim number of layers, that is the height of the tree T;
- which types of nodes can be used;
- the pool $\mathbb{F}$ of candidate feature extractors that can be used as *leaf nodes*;
- a range and a step for each meta-parameter must be given to perform the greedy search;
- the training meta-parameters for MLP networks are the same for all the nodes.

The *BPConf* process also requires the definition of the concept of error map: let $N_i \in T$ be an *inner node* and I an image, so that the

node N_i is trained to learn a function h^* such that $h^*(I) = M_I^{N_i}$. The error map for the image I and the node N_i is defined as follows:

$$E_I^{N_i} = |M_{GT} - M_I^{N_i}| \tag{5.1}$$

where the subtraction is pixel-wise so that $E_I^{N_i}$ actually is a 2-dimesional map with the same size of I resampled to w, given w the resample size of N_i.

BPConf basically consists in two steps, shown in Figure 5.1, which are iterated: a greedy search for the best node to be added to T, called *first step*, followed by a greedy search for a child node specifically trained to compensate the errors committed by the previous node, called *second step*.

First step - This step tries to add a new root node N to the existing MNOS T along with the set of feature $F_N \in \mathcal{P}(\mathbb{F})$. N is selected by performing a greedy search for all the possible types of nodes and feature extraction combinations. A grid search is performed in the parameters space by following the previous constraints. Every time a new tree $\tilde{T}$ is created by adding N as parent of the root node of T and N is trained using both the feature patterns from F_N and the likelihood map produced by T for the images in D_{train}. The best tree $\bar{T}$ among all the attempts is selected and it replaces T. The first time this step is executed, $T = \varnothing$ so the best N and becomes the first node in T (Fig. 5.1(b)). The following times, this step just add a new node as root to T.

Second step - It tries to add a new node as child node of the current root node (Fig. 5.1(c)). Let N_{root} be the root node of the MNOS tree T, then the error maps $E_{I^{(k)}}^{N_{root}} \forall I^{(k)} \in D_{train}$ are calculated.

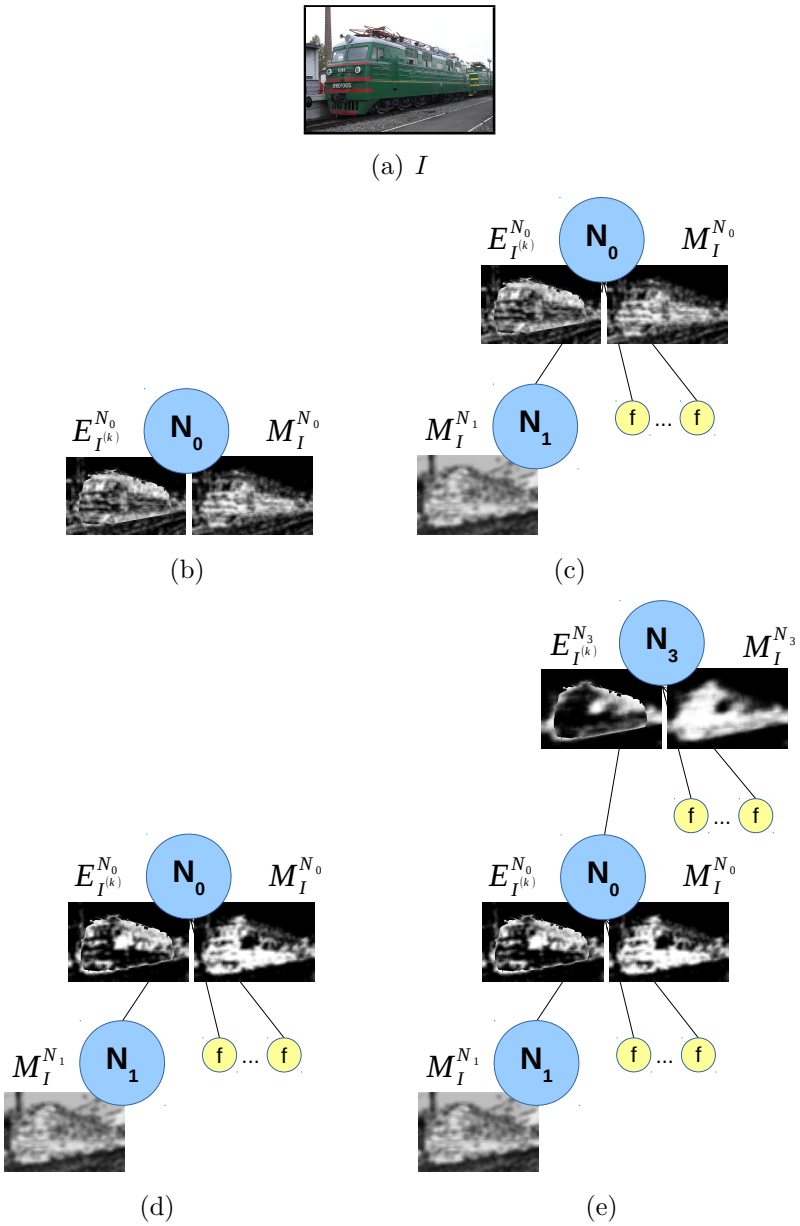


Figure 5.1: *Steps performed by BPConf. (b) A greedy search is performed for the first node N_0, and its error map is computed; (c) a new node N_1 is trained to help N_0 improve its performance; (d) N_0 is retrained; (e) a new iteration starts by adding a new node N_3 as root node.*

For each image $I^{(k)}$ a new node N is trained on the points where $E^{N_{root}}_{I^{(k)}}$ is higher in order to specialize the hypothesis learnt by N to correct the errors commited by N_{root}. A greedy search like the one for the *first step* over the possible configurations for N and for choosing F_N is performed with an additional constraint: N can only be a node of type Sw because it performs a pixel level prediction, as discussed in Section 3.1.3, so it is possible to constrain the sliding windows to prefer focusing on higher error regions. It would not be possible to do it with Seg nodes. Every attempt a new tree $\tilde{T}$ is created by adding N as child node to the root node of T, which is retrained with the additional input provided by N. The best tree $\bar{T}$ among all the attempts is selected and it replaces T if and only if average $IoU_{\bar{T},D_{val}} >$ average $IoU_{T,D_{val}}$, that is the average IoU described in Equation 3.5 for all the images in D_{val}. This step is shown in Figure 5.1(c) while Figure 5.1(d) shows that once the root node is retrained, both its error map and its likelihoood map improve.

The two steps are repeated until T has the required number of layers or if no nodes are added during an iteration.

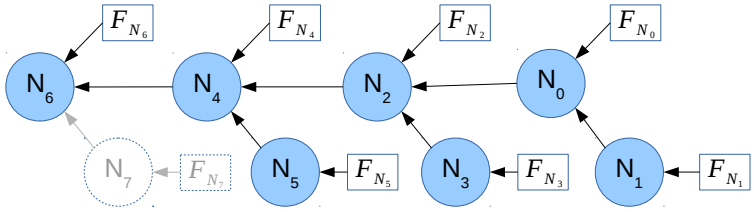
The interpretation of the error map E^N_I is twofold: it is considered a probability distribution over the points of I and the sliding window positionings during the training of the MLP are sampled from that distribution, but it can also be considered a saliency map that focus on the image regions for which N fails to assign a good likelihood estimate, thus guiding the positioning of the the neural network receptive field, situation that reminds a biological visual system.

BPConf creates heavy unbalanced trees like the one in Figure 5.1(e) that are very similar to a cascade of classifiers; each *inner node* has at most two child nodes (excluding the feature extractors), so it receives at most two bottom-up signals, aside from the feature patterns produced by the *leaf nodes*:

- the segmentation proposal from the previous layer;

Table 5.1: *Average Mean Square Error (MSE) of the likelihood map produced by each node before adding a node to help it to correct its error and after using BPConf. It has been measured on the validation set for the VOC2010 class "train" on a network with the topology shown in the figure below.*

		MSE		
Node	**Type**	**w/o node**	**w/ node**	**IoU**
N_0	*Sw*	256	198	0.18
N_2	*Sw*	166	141	0.38
N_4	*Seg*	144	128	0.52
N_6	*Seg*	122	131	0.57



- eventually a likelihood map from a node explicitly trained to help correct the errors.

Table 5.1 shows that adding a child node selected by the *second step* helps to decrease the error committed by the node itself. A MNOS has been generated by *BPConf* using the images from the "train" class of the VOC2010 dataset [41]. Given a node $N \in T$ and a validation set $D_{val} \subset D$, the average MSE is computed as follows:

$$\text{Average}MSE(N, D_{val}) = \frac{1}{|D_{val}|} \sum_{k=1}^{|D_{val}|} \sum_{p=1}^{|I^{(k)}|} |M_{GT}^{(k)}(p) - M_{I^{(k)}}^{N}(p)|^2 \quad (5.2)$$

The Average MSE of a node N_i is measured on the validation set before executing the *second step* after: as proved by Table 5.1, every

time a child node is added, the average MSE of its parent node decreases. This is not obvious as the criterion for adding a child node is based on a different accuracy measure, the IoU, as it is more suitable to evaluate image segmentation tasks; on the other hand, the MSE is more appropriate to measure the distance of a single classifier output (the likelihood map $M_I^{N_i}$) with respect to the actual target (the ground truth mask M_{GT}), even if the two measures actually seems to be correlated. From the table it is possible to notice some interesting situations. Adding N_4 alone as root node actually worsened the performances of the ensemble but by training a child node to compensate its errors the MSE became lower and the node has been kept as the IoU decreased. N_4 is a *Seg* node while its child is a *Sw* node: as discussed in Section 3.1.4, there is a nice synergy between these two types of nodes when one or more hypothesis from *Sw* nodes are refined by a *Seg* node while a *Seg* node alone may have difficulties in presence of complex backgrounds or high contrasts: this can be solved introducing a bottom-up signal like the likelihood maps computed by *Sw* nodes, which are similar to the one of saliency detection. The child node N_7 is not added as it did not help to reduce the errors commited by N_6 which is kept anyway as the IoU increases. In Figure 5.1(c) and 5.1(d) it is possible to look at how both the error map and the likelihood map produced by a node vary as the child node is added.

Table 5.2 expands Table 3.2 by showing the results obtained by MNOS ensembles with at most 4 layers configured by *BPConf* on the VOC2010 dataset [41], figure-ground segmentation variant. This results are directly comparable to the ones in Table 3.2, which are reported here for convenience. The performance of a MNOS configured by *BPConf* overcomes both the MNOS ensembles and the MNOS with the GrabCut refinement used in Section 3.1.3.

Table 5.2: *Results on the VOC2010 Figure-ground variant for hand tuned MNOS, MNOS with GrabCut refinement and MNOS configured by BPConf.*

Class	MNOS	GrabCut	*BPConf*
Airplane	0.36	0.56	0.56
Bicycle	0.15	0.13	0.24
Bird	0.24	0.37	0.28
Boat	0.31	0.36	0.41
Bottle	0.21	0.19	0.27
Bus	0.51	0.57	0.65
Car	0.37	0.38	0.42
Cat	0.37	0.39	0.46
Chair	0.05	0.10	0.15
Cow	0.41	0.54	0.56
Dining Table	0.26	0.26	0.30
Dog	0.35	0.38	0.42
Horse	0.31	0.42	0.52
Motorbike	0.51	0.49	0.57
Person	0.33	0.36	0.36
Potted plant	0.19	0.24	0.35
Sheep	0.31	0.32	0.56
Sofa	0.26	0.28	0.36
Train	0.50	0.52	0.57
TV monitor	0.18	0.27	0.31

Average IoU	
Küttel and Ferrari [42]	0.48
Carreira and Sminchisescu [43]	0.34
Rosenfeld and Weinshall [44]	0.46
MNOS Average	0.31
MNOS + GrabCut Average	0.36
MNOS (BPConf) Average	0.42

This also allows to obtain better segmentation accuracies while reducing the computational time in generalization, as MNOS with GrabCut segmentations need 500ms in average per image while MNOS configured by *BPConf* obtains better results in less than 100ms.

5.2 High Entropy Ensembles for Figure-ground Segmentation

The configuration algorithm for the MNOS *BPConf* [68] discussed in Section 5.1 provides a good strategy for configuring both the topology and the meta-parameters of a MNOS. MNOS ensembles created by *BPConf* obtain good results on the VOC2010 standard dataset and, above all, prove how interaction between algorithms help improving the performance when combined together to perform a single task. Anyway, *BPConf* has some limitations as the ensemble topology is similar to a cascade of classifiers and do exploit all the possible interactions offered by a tree topology. This simplification help to reduce the cumputational cost of *BPConf*, which is very demanding as for each node to add i perform a grid search in the meta-parameter space and for all the combination of available feature extractor nodes, retraining the neural network every time; it follows that the computational time needed to configure a MNOS strictly depends on the range and step for each meta-parameter value, the number of candidate feature extractors and also the cardinality of the dataset.

It would be great to find a strategy that do not need greedy searches in such spaces, thus allowing to spend more resources to focus on the search of more sophisticated interactions between the components of the ensemble. This is done by a new configuration algorithm called High Entropy Ensembles [69] which, starting from the strength of *BPConf*, solves the previous issues by "injecting random-

ness" in the optimization process, also allowing the definition of a more general framework to combine any figure-ground segmentation algorithm into an ensemble.

5.2.1 Problem Formulation

As presented in Section 3.1 the components of a MNOS ensemble are the *inner nodes*, that are classifiers, and *leaf nodes*, which are feature extractors that calculate feature patterns from the input images for the classifiers. So far, a MNOS used just two type of nodes, which are paradigmatic algorithms for two possible strategies: pixel-wise classification based on the intensity values distribution and image regions classification based on local features. Both these algorithms learn the same type of function described in Section 3.1.1, which makes them calculate a likelihood map by taking in input a set of feature patterns and eventually likelihood maps from nodes in the previous layer.

In order to cast the MNOS to a more general framework for figure-ground segmentation algorithms combination, a new formulation for the problem must be given. Let $\mathbb{A}$ be a set of figure-ground segmentation algorithms and $\mathbb{F}$ a set of feature extractor algorithms. Each algorithm $a \in \mathbb{A}$ must comply to the following interface:

- **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 $\mathbb{A}$;

- **output**: a single gray-scale image in which the intensity level assigned to each pixel represents the probability that it belongs to foreground.

and each $f \in \mathbb{F}$ comply to the following:

- **input**: the image I given in input to the ensemble;
- **output**: a feature pattern calculated from I.

The previous constraints are reasonable as every figure-ground segmentation algorithm can be easily modified to produce such a segmentation map and also to take in input a custom set of features, similarly to what CCM does [6] (see Sec. 2.3): by considering the algorithms in $\mathbb{A}$ as black boxes there is not the problem of understanding the details of the specific algorithms that are employed. The objective is the definition of a model combination algorithm for the algorithms in $\mathbb{A}$ using the features calculated from the feature extractors $\mathbb{F}$. Unlike CCM, the algorithms in $\mathbb{A}$ perform all the same task and each component of the ensemble is organized in a tree structure that is learnt during the training procedure.

5.2.2 Building phase

Let D be a dataset like the one described in Equation 3.4 and $D_{train} \cup D_{val} = D$ such thet $D_{train} \cap D_{val} = \varnothing$ are respectively a training and a validation set. The *building phase* is the procedure for building a segmentation tree T by combining a set of randomly configured algorithms from $\mathbb{A}$, called nodes like in the MNOS. The whole method is driven by the maximization of a goodness function $G(T, D_{val})$ computed as the average IoU (described in Eq. 3.5) of T for the validation set $D_{val} \subset D$:

$$G(T, D_{val}) = \frac{1}{|D_{val}|} \sum_{k=1}^{|D_{val}|} IoU_{T,I^{(k)}} \tag{5.3}$$

Equation 5.3 is defined to take into account pixel-wise precision and recall and measure the performance of the ensemble T in relation to the high level task it is facing, that is figure-ground segmentation,

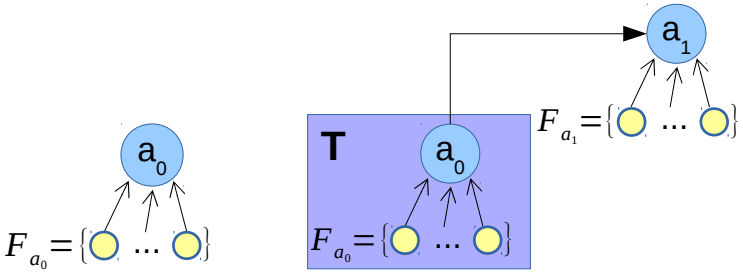
independently from the nature of the algorithms in $\mathbb{A}$. The *building phase* consists in an initialization, the *base step*, followed by an iterative procedure.

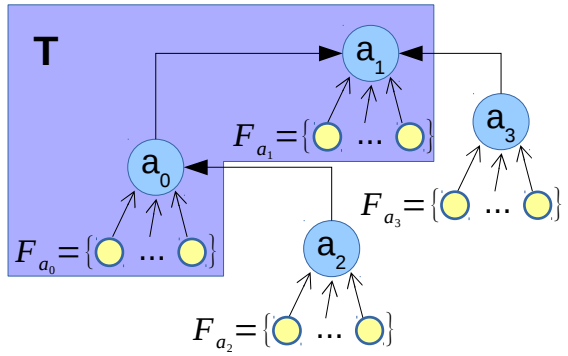
Base step - the goal of this step is to create the first node of the tree T, as shown in Figure 5.2(a). We randomly select a figure-ground segmentation algorithm $a \in \mathbb{A}$ along with a random set of feature extractors $F_a \in \mathcal{P}(\mathbb{F})$. After that, the selected node a is trained using the feature patterns produced by the feature extractors in F_a. The previous actions are repeated n times until $\bar{a}$ is identified as the node that maximizes $G(a, D_{val})$ over all the n generated nodes. Finally, a becomes the first and only node in T.

In order to extend T, after the *base step* a *bottom-up step* is performed, 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 Figure 5.2(b). Similarly to the *base step*, a random selection of a figure-ground segmentation algorithm $a \in \mathbb{A}$ is performed along with a random set of feature extractors $F_a \in \mathcal{P}(\mathbb{F})$. Then, a new tree $\tilde{T}$ is created by adding a as parent of the root node of T. This means that the output produced by T is given as input to a along with the set of patterns generated by the feature extractors in F_a. The node a is trained using the patterns produced by its feature extractors in F_a and the map generated by T. The previous actions are repeated n times, always starting from the original tree T. Finally, $\bar{T}$ is identified as the tree $\tilde{T}$ that maximizes $G(\tilde{T}, D_{val})$ over all the n generated trees. The original tree T is replaced by $\bar{T}$ if and only if $G(\bar{T}, D_{val}) - G(T, D_{val}) > \epsilon$.

Top-down step - the goal of this step is to add a child node to each existing node in T, as shown in Fig. 5.2(c). A new tree $\bar{T}$ is created by adding a as a child of $a' \in T$.





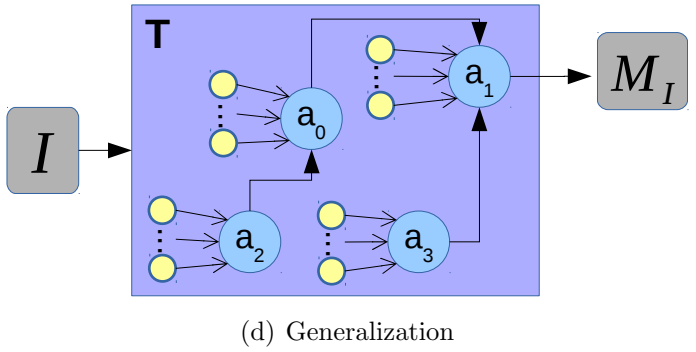


Figure 5.2: *HEE building phase. The procedure starts with the (a) base step that creates the first node of T. Then the (b) bottom up step and the (c) Top-down step are repeated until no node is added to T and the final ensemble is used in (d) generalization.*

It means that the output produced by a is given as input to a' along with the set of patterns generated by the feature extractor algorithms in $F_{a'} \in \mathcal{P}(\mathbb{F})$. The nodes of $\bar{T}$ are recursively retrained starting from a all the way up to the root. The previous actions are repeated at most n times (always starting from the original tree T) until $G(\bar{T}, D_{val}) - G(T, D_{val}) > \epsilon$. If this condition is satisfied then $\bar{T}$ becomes T. Note that the procedure iterates 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 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. The final ensemble can be used to produce the segmentation map for a given image I as summarized in Figure 5.2(d): the segmentation map M_I for an image I is generated by T by giving I as input to all the feature extractors $f \in F_{a_i}\ \forall a_i \in T$. The feature patterns generated by the extractors in F_{a_i} are exploited by the node $a_i \in T$ to produce a segmentation map $M_I^{a_i}$ that is given as input to its parent. This procedure is repeated until a segmentation map M_I is produced by the root node of T. The final binary segmentation map is obtained by thresholding M_I with a threshold level equals to half the maximum intensity value of the map.

5.2.3 Analysis of the building phase

The idea of using randomness is not new as other works obtain good results by injecting randomness in their models [70, 71]. Our method is far from boosting based approaches [4, 12] or CoBE [13, 14] (see Sec. 2.2) 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 [5, 10], 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 [6]. 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; The tree-based structure has been maintained from the MNOS because, as investigated in [8, 9] and Section 5.2.4, 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 CoBE, CCM, model averaging and other linear model combination methods (see Section 2).

The following experiments, performed on the Weizmann Horses dataset [72], prove the effectiveness of the proposed approach by showing that: (i) randomly configured nodes achieve poor results when used individually, (ii) better performances 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. When not explicitly stated otherwise $\mathbb{A} = \{Sw, Seg\}$, the two algorithms used as MNOS nodes discussed in Section 3.1; the reason is twofold: to allow a comparison with the *BPConf* configuration strategy and because this two algorithms represents two common solutions to address segmentation tasks, as mentioned before, so they are enough representative. On the other hand, $\mathbb{F}$ is the set of feature extraction algorihtms described in Chapter 4. We also set $n = 12$ and $\epsilon = 0.05$ after trying different ranges of values. As previously discussed the lower n is the faster the procedure becomes bacause less attempts are

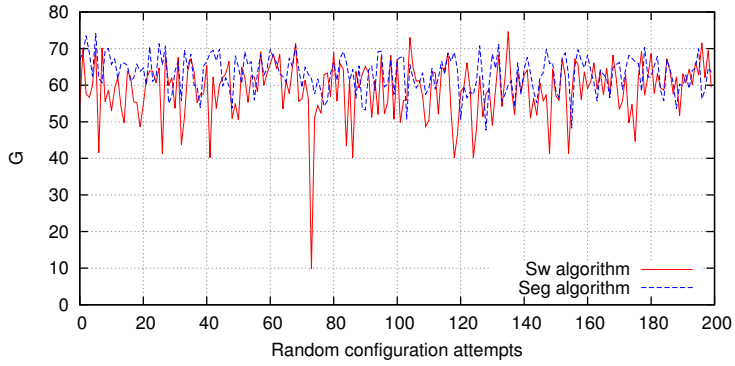


Figure 5.3: *Goodness value G for the Weizmann Horses test set when using single nodes generated by randomly configuring* 200 *times the algorithms Sw and Seg.*

performed at each step and this value is a good compromise; moreover, a low value for ϵ allows the tree to easily grow as just a small improvement is needed to add a node, thus allowing to build bigger ensembles. 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*.

First, a set of nodes is generated by randomly configuring 200 times each algorithm in $\mathbb{A}$; after that, the goodness values they individually achieve on test set are computed. The results we obtain are shown in Figure 5.3. The randomly generated nodes yield poor mean results μ when used individually and that the standard deviation σ of their goodness values is small: $\mu_{Sw} = 58.90$, $\sigma_{Sw} = 7.87$,

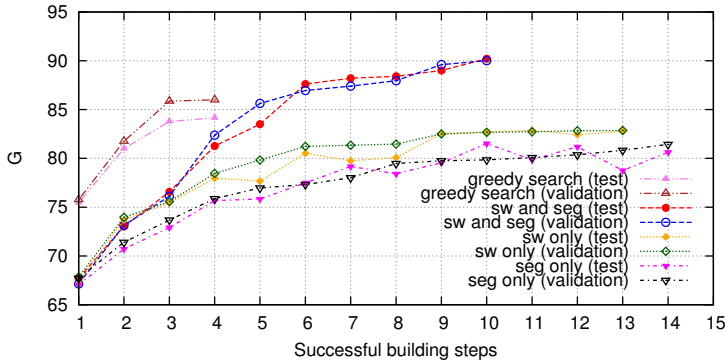


Figure 5.4: *Comparison between goodness achieved for the Weizmann Horses test set after each successful step of the building phase while varying the algorithms in* $\mathbb{A}$ *and the parameters selection technique. The building phase for the mixed Sw and Seg tree terminates after just* 10 *steps and achieves significantly better results both on test and validation than the ones obtained using an optimal parameter initialization approach or just one type of segmentation algorithm.*

$\mu_{Seg} = 63.13$, $\sigma_{Seg} = 5.95$.

In order to prove that it is possible to obtain better results if some of these nodes are combined using the HEE strategy, a second experiment in which we execute the *building phase* to generate a tree T is performed; after each successful step the goodness of T for both the validation and the test sets is reported. Results are shown in Figure 5.4. Taking into account that each successful step adds a new node to T, it is possible to 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. 5.2.2). The latest is more interesting, because it is in accor-

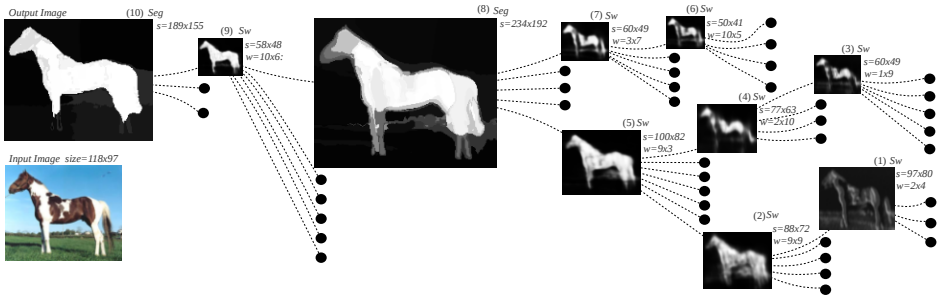


Figure 5.5: *An example showing the segmentation tree that outperforms the state-of-the-art results for the Weizmann Horses dataset. The different nodes compensate each others' errors: Seg nodes produce accurate segmentations near the contours but lose parts of the object, Sw nodes produce blurred segmentations but capture the whole object. The segmentation map for the given input image is produced in less than one second.*

dance with the assumption that different segmentation algorithms may commit different mistakes [3] 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 Figure 5.4, 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 MSE (Eq. 5.2) produced by the nodes of the tree of Figure 5.5 on the test set. The results obtained are presented in Figure 5.6. To make the graph more readable, we compute the average MSE values only for the nodes lying on the path that goes from the *leaf node* labeled in Figure 5.5 as (1) to the root (10). Results show that the mean square error de-

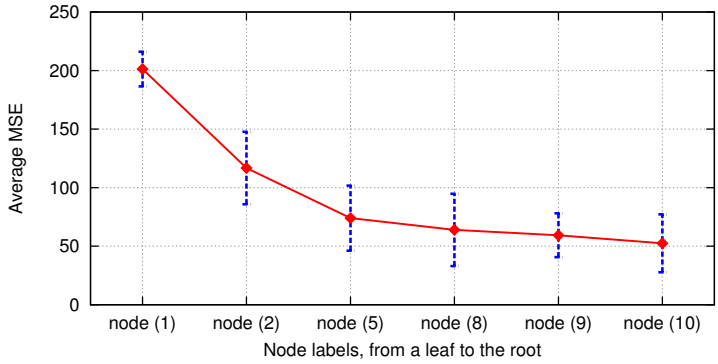


Figure 5.6: *Average mean square error achieved for the Weizmann Horses test set by the nodes of the model showed in Fig. 5.5, on a path from a leaf to the root. The reported error bars represent the standard deviation of the mean square error. The error decreases as we climb the ensemble toward its top node.*

creases 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.

As a side note, it is possible to notice that the last layers of the ensemble of Figure 5.5 are *Seg* nodes: they have been selected automatically by the *building phase* because it finds out that they usually produce better results than *Sw* nodes when used in the layers close to the root and they produce a crisp and precise segmentation of the details and the border of the object in the images, as discussed in Section 3.1.4.

Another experiment is performed to provide a comparison between the random and the fixed parameters selection approaches. In

the fixed parameters selection approach we perform a greedy 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 [73, 6], and *BPConf* (Sec. 5.1). The greedy 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. 5.4. The *building phase* using the greedy search approach terminates after adding just 4 nodes to the ensemble, which means that the *building phase* becomes unable to add any new node very soon. This does not surprise since, by exhaustively looking for the best node at each step, it is like approximating other model combination frameworks (such as CCM or PSL); the ensembles generated by those methods are usually not deep, mostly to prevent overfitting the training data. In fact such approaches are very prone to overfit as they tend to get stuck into local minima of the function they are optimizing and need regularization techniques. On the other hand, randomness injection is a powerful regularization mechanism that allow to solve such problems.

Additional experiments were performed to determine whether all the nodes in the trees contribute toward the final predictions. A significant drop in performance has been experienced when *leaf nodes* are recursively removed without retraining the nodes in the trees not affected by the change; the same behavior was observed when removing entire branches.

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 [61] (LUV, gradient histogram and gradient magnitude) widely used by state-of-the-art rigid object detection algorithms. This proves that, even tough the proposed model is heavily random-based, it is able to build optimal segmentation ensembles.

The time required to complete each successful step of the *building phase* for the model presented in Figure 5.5 is exponential in the number of previous successful steps. In fact, as the *building phase* approaches 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 and this is the major drawback of High Entropy Ensembles. The *building phase* for the model of Figure 5.5 lasted approximately 3 days on an average PC and single thread. Anyway, there are margins to speed up the computation by parallelizing the computation. The time required by a tree to perform the segmentation of an image depends on the computational complexity of its nodes.

5.2.4 Comparison with standard datasets

HEE have been compared with other state-of-the-art algorithms and model combination methods. For all the experiments, the same set of parameters described in Sec. 5.2.2 is used. The comparisons are carried out for the following datasets: Weizmann Horses [72], Oxford Flower 17 [74], INRIA Graz-02 [75] and the f/g variant of VOC2010 [42].

The segmentation performances for the first two datasets were measured using the same metrics of [76]:

$$S_a(T, I) = \frac{1}{|I|} \sum_{p}^{|I|} I[M_I(p) = M_{GT}(p)] \tag{5.4}$$

$$S_o(T, I) = IoU_{T,I} \tag{5.5}$$

S_a is the overall pixel accuracy and S_o is the foreground overlapping ration, that has been previously defined as IoU by Equation 3.5.

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 [42]. For the figure-ground variant of the VOC2010 dataset, the pixel-wise IoU as in [42]. In each experiment, comparisons with AdaBoost [5] and CoBE [14, 4] (Sec. 2.2), BMA [16] (Sec. 2.1) and CCM [6] (Sec. 2.3) are performed. For the Weizmann Horses dataset, a comparison with the Auto-context cascade [7] (Sec. 2.4) is performed too.

AdaBoost was applied to the algorithms in $\mathbb{A} = \{Sw, Seg\}$ in the following way: a family of figure-ground segmentators $\mathbb{H}$ has been built using the same greedy search strategy previously used for configuring optimal nodes: at most 20 optimal classifiers $h_i \in \mathbb{H}$ are selected and combined by computing the weight distribution over the pixels in the training set. Similarly, we obtain the results for CoBE (nodes are boosted using AdaBoost as in [4]) 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.

Figures 5.7 and 5.8 respectively show good and bad segmentation results obtained on the INRIA Graz-02 dataset, which is the most difficult dataset, similarly to VOC2010 as while the formers contain images where the object is always in the center of the image, well visible and separated from the background, the latters present occlusions, complex lighting conditions and very variable scenes.

Method	S_a (%)	S_o (%)
Küttel et al. [42]	94.7	/
Bertelli et al. [76]	94.6	80.1
Seyedhosseini et al. [77]	95.4	/
AdaBoost [5]	90.0	72.9
CCM [6]	89.3	79.6
BMA [16]	77.1	58.9
CoBE [4]	90.8	76.0
LHEE	87.1	72.5
HEE	**98.2**	**90.2**

Table 5.3: *Results on Weizmann Horses dataset.*

Method	S_a (%)	S_o (%)
Nilsback et al. [74]	/	94.0
Bertelli et al. [76]	97.7	92.3
Chai et al. [78]	/	90.4
AdaBoost [5]	93.1	85.5
CCM [6]	86.3	84.5
BMA [16]	87.3	81.0
CoBE [4]	95,8	90,6
LHEE	89,6	87,6
HEE	**98.1**	**96.1**

Table 5.4: *Results on Oxford Flower 17 dataset.*

Method	cars	people	bikes	avg.
Marszałek et al. [75]	53.8	44.1	61.8	53.2
Küttel et al. [42]	74.8	66.4	63.2	68.1
Fulkerson et al. [79]	72.2	66.3	72.2	70.2
AdaBoost [5]	60.1	48.6	63.0	57.2
CCM [6]	62.6	55.9	72.8	64.4
BMA [16]	55.4	53.4	65.3	56.0
CoBE [4]	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**

Table 5.5: *Results on INRIA Graz dataset.*

Average IoU	
Küttel and Ferrari [42]	0.48
Carreira and Sminchisescu [43]	0.34
Rosenfeld and Weinshall [44]	0.46
MNOS Average	0.31
MNOS + GrabCut Average	0.36
MNOS (BPConf) Average	0.42
HEE	0.56

Table 5.6: *Results on VOC2010 dataset.*

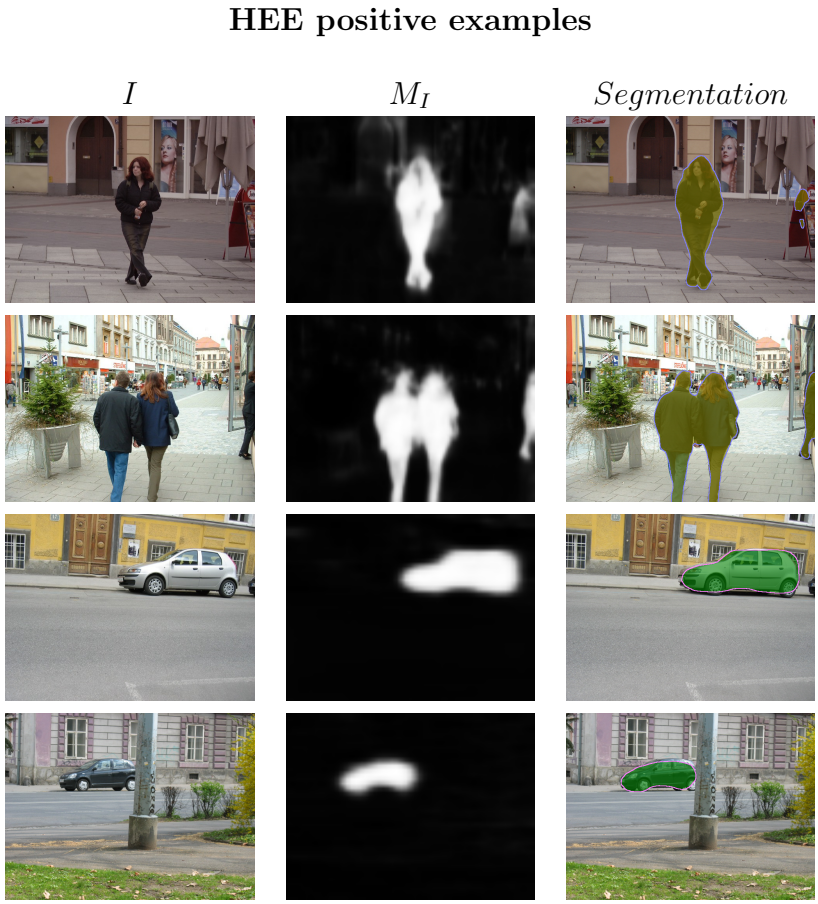


Figure 5.7: *Examples of good segmentation results for images from different classes of INRIA Graz-02 dataset. M_I denotes the soft figure-ground segmentation maps generated by the root node of HEE for the given input image I. The final Segmentation is obtained by thresholding M_I to half its maximum intensity value.*

Figure 5.8: *Examples of bad segmentation results for images from different classes of INRIA Graz-02 dataset. M_I denotes the soft figure-ground segmentation maps generated by the root node of HEE for the given input image I. The final Segmentation is obtained by thresholding M_I to half its maximum intensity value.*

6
Conclusion

In this work the problem of object segmentation and figure-ground segmentation has been addressed by employing ensembles of algorithms. In literature this approach is not new as several models face computer vision problems and image segmentation by combining different algorithms. The nature of this combination can be summarized by two different approaches: the use of rejection rules, usually in cascaded classifiers, of interactions, like graphical models.

This work started by discussing an ensemble called Multi-Net for Object Detection, devised for object detection. It is a tree-like graphical model that used MLP neural networks with sliding windows approach to estimate the location of objects of interest in images. It has been modified for object segmentation by adding a new type of node to the graphical model that is more suitable for this task, creating

the Multi-Net for Object Segmentation. This solution was motivated by the industry and it has been tested on a dataset composed by images of commercial offers sold online by websites, which represent clothes, obtaining good results. Anyway once tested on a standard and challenging dataset, the results were not acceptable because of the complexity of natural scenes compared to the previous images.

Actually, the main problem of the MNOS is that the creation of a graphical model topology by hand is not the most effective way. This problem has been addressed and a new strategy for automatic configuration for a MNOS ensemble has been discussed and called *BPConf*. It iteratively builds a tree very similar to a cascade of algorithms but adding nodes specifically trained to correct the errors committed at each iteration. This solution led to instantiate MNOS trees that perform better on the same standard dataset used before.

The previous solution is based on greedy searches for the optimal nodes to add to the graphical model at each iteration: for computational complexity reasons, the amount of interactions defined by *BPConf* is very limited and, as *BPConf* itself suggests, a good configuration of a structure where algorithms interact allows to improve the results of such kind of graphical models. This is the reason why a new model combination framework, called HEE, has been devised: MNOS has been generalized as ensemble of any kind of algorithms and a building strategy has been presented, which heavily relies on randomness to address the problem of computational complexity due to the huge meta-parameter space of this problem, like several other algorithms proposed in literature.

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. When the algorithms involved in the model combination procedure are used alone, they have poor performances on average

but, once they are combined, they always perform better. Moreover, the performaces of both a greedy search approach and the HEE model combination framework are compared: it results that the former approach is far less capable to build an affective ensemble. On the other hand, the HEE building phase is not a deterministic procedure as it heavily relies on randomness and the advantage is twofold: a better computational efficiency allows to look for more complex interactions between the components and the building phase do not get stuck in local maxima of the goodness function. Moreover, HEE prove that an ensemble based on interactions and composed by different algorithms perform better than an ensemble built using the same strategy but using only one algorithm as component, such as Auto-context. All these results are proved by using standard figure-ground segmentation datasets widely employed in literature and, for each of them, HEE beats the current state-of-the-art results.

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. The most important result proved by High Entropy Ensembles is that in ensemble of algorithms, by preferring interactions instead of rejection rules it is possible to achieve far better results if the computational time is not an issue and, by working on reducing the computational complexity of such kind of algorithms, it would be possible in the future to obtain a sensible improvement in the field of model combination algorithms.

Colophon

Credits for this Latex template go to Moreno Carullo.

Most of the software developed uses C++ and C# while early attempts were developed using Matlab.

The computer vision library OpenCv [47] was widely employed in the C++ and C# code.

For linear algebra calculus, the library Eigen [80] was used.

Acronyms

BMA Bayesian Model Averaging.

CBIR Content Based Image Retrieval.

CCM Cascaded Classification Models.

CoBE Cascades of Boosted Ensembles.

DoG Derivatives of Gaussian.

GMM Gaussian Mixture Model.

HEE High Entropy Ensembles.

HOG Histogram of Oriented Gradients.

IoU Intersection over Union.

LoG Laplacian of Gaussian.

MLP Multi-layer Perceptron.

MNOD Multi-net for Object Detection.

MNOS Multi-net for Object Segmentation.

MSE Mean Square Error.

RPROP Resilient Backpropagation.

Bibliography

[1] M. Livingstone, *Vision and art: the biology of seeing.* New York: Harry N. Abrams, 2002.

[2] R. Szeliski, *Computer vision algorithms and applications.* Springer, 2011.

[3] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[4] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, 2004.

[5] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[6] G. Heitz, S. Gould, A. Saxena, and D. Koller, "Cascaded classification models: Combining models for holistic scene understanding," in *International Conference on Neural Information Processing Systems*, 2008.

[7] Z. Tu, "Auto-context and its application to high-level vision tasks." in *International Conference on Computer Vision and Pattern Recognition*, 2008.

[8] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. H. Witten, "Using model trees for classification," *Machine Learning*, vol. 32, no. 1, pp. 63–76, 1998.

[9] B. Zenko, "Is combining classifiers better than selecting the best one?" *Machine Learning*, vol. 54, no. 2, pp. 255–273, 2004.

[10] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[11] Z. Tu and X. Bai, "Auto-context and its application to high-level vision tasks and 3d brain image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1744–1757, 2010.

[12] B. Wu and R. Nevatia, "Simultaneous object detection and segmentation by boosting local shape feature based classifier," in *International Conference on Computer Vision and Pattern Recognition*, 2007.

[13] M. J. Saberian and N. Vasconcelos, "Boosting classifier cascades," in *International Conference on Neural Information Processing Systems*, 2010.

[14] T. Susnjak, A. L. C. Barczak, and K. A. Hawick, "Adaptive cascade of boosted ensembles for face detection in concept drift," *Neural Computing and Applications*, vol. 21, no. 4, 2012.

[15] T. M. Mitchell, *Machine Learning*, 1st ed. McGraw-Hill, Inc., 1997.

[16] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: A tutorial," *Statistical Science*, vol. 14, no. 4, 1999.

[17] M. Opper and D. Haussler, "Calculation of the learning curve of bayes optimal classification algorithm for learning a perceptron with noise," in *In Computational Learning Theory: Proceedings of the Fourth Annual Workshop*.

[18] D. Haussler, M. Kearns, and R. E. Schapire, "Bounds on the sample complexity of bayesian learning using information theory and the vc dimension," *Machine Learning*, vol. 14, 1994.

[19] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," *Machine Learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[20] P. Domingos, "Bayesian averaging of classifiers and the overfitting problem," in *International Conference of Machine Learning*, 2000.

[21] T. P. Minka, "Bayesian model averaging is not model combination," Tech. Rep., 2002.

[22] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[23] M. Kearns, "Thoughts on hypothesis boosting," Tech. Rep., 1988.

[24] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *International Conference on Computer Vision and Pattern Recognition*, 2001.

[25] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real adaboost," in *IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.

[26] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl, "Aggregate features and adaboost for music classification," *Machine Learning*, vol. 65, no. 2-3, pp. 473–484, 2006.

[27] O. M. Mozos, C. Stachniss, and WolframBurgard, "Supervised learning of places from range data using adaboost," in *IEEE International Conference on Robotics and Automation*, 2005.

[28] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Annals of Statistics*, vol. 26, no. 5, pp. 1651–1686, 1997.

[29] Z. Tu and X. Bai, "Auto-context and its application to high-level vision tasks and 3d brain image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1744–1757, 2010.

[30] I. Gallo and A. Nodari, "Learning object detection using multiple neural netwoks," in *International Conference on Computer Vision Theory and Applications*, 2011.

[31] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area v2," in *Advances in Neural Information Processing Systems 20*, 2008.

[32] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, vol. 20, no. 11, pp. 1254–1259, 1998.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning internal representations by error propagation*, 1986.

[34] M. Riedmiller and H. Braun, "Rprop - a fast adaptive learning algorithm," Proc. of ISCIS VII), Universitat, Tech. Rep., 1992.

[35] S. Esugasini, M. Y. Mashor, N. A. M. Isa, and N. H. Othman, "Performance comparison for MLP networks using various back propagation algorithms for breast cancer diagnosis," in *Knowledge-Based Intelligent Information and Engineering Systems*, 2005, vol. 3682, pp. 123–130.

[36] F. Li, J. Carreira, and C. Sminchisescu, "Object recognition as ranking holistic figure-ground hypotheses," in *IEEE International Conference on Computer Vision and Pattern Recognition*, 2010.

[37] S. Albertini, I. Gallo, M. Vanetti, and A. Nodari, "Learning object segmentation using a multi network segment classification approach," in *International Conference on Computer Vision Theory and Applications*, 2012.

[38] J. Hartigan and M. Wang, "A k-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.

[39] M.-K. Hu, "Visual pattern recognition by moment invariants," *Information Theory, IRE Transactions on*, vol. 8, no. 2, pp. 179–187, 1962.

[40] I. Gallo, A. Nodari, and M. Vanetti, "Object segmentation using multiple neural networks for commercial offers visual search," in *Engineering Applications of Neural Networks*, 2011.

[41] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[42] D. Küttel and V. Ferrari, "Figure-ground segmentation by transferring window masks," in *International Conference on Computer Vision and Pattern Recognition*, 2012.

[43] J. Carreira and C. Sminchisescu, "Constrained parametric min-cuts for automatic object segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1312–1328, 2012.

[44] A. Rosenfeld and D. Weinshall, "Extracting foreground masks towards object recognition," in *International Conference on Computer Vision*, 2011.

[45] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut": Interactive foreground extraction using iterated graph cuts," vol. 23, 2004, pp. 309–314.

[46] D. Reynolds, "Gaussian mixture models," *Encyclopedia of Biometrics*, pp. 659–663, 2009.

[47] G. Bradski, "The opencv library," *Dr. Dobb's Journal of Software Tools*, 2000.

[48] A. Nodari, I. Gallo, M. Vanetti, and S. Albertini, "Color and texture indexing using an object segmentation approach," *Engineering Intelligent Systems*, vol. 20, no. 1-2, pp. 47–57, 2012.

[49] Y. Zhong and A. K. Jain, "Object localization using color, texture and shape." *Pattern Recognition*, vol. 33, no. 4, pp. 671–684.

[50] A. Nodari and I. Gallo, "Image indexing using a color similarity metric based on the human visual system," in *International Conference on Machine Vision, Image Processing, and Pattern Analysis*, 2011.

[51] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[52] A. Nodari, M. Ghiringhelli, A. Zamberletti, M. Vanetti, S. Albertini, and I. Gallo, "A mobile visual search application for content based image retrieval in the fashion domain," in *10th International Workshop on Content-Based Multimedia Indexing*, 2012.

[53] M. Vanetti, I. Gallo, and A. Nodari, "Gas meter reading from real world images using a multi-net system," *Pattern Recognition Letters*, vol. 34, no. 5, pp. 519–526, 2013.

[54] B. Julesz, "Textons, the elements of texture perception, and their interactions," *Nature*, vol. 290, no. 5802, pp. 91–97, 1981.

[55] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and texture analysis for image segmentation," *International Journal of Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[56] S. Zhu, C. Guo, Y. Wang, and Z. Xu, "What are textons?" *International Journal of Computer Vision*, vol. 62, no. 1-2, pp. 121–143, 2005.

[57] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *International Conference on Artificial Intelligence and Statistics*, 2011.

[58] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area v2," in *Advances in Neural Information Processing Systems*, 2008.

[59] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[60] J.-B. Thomas, P. Colantoni, and A. Trémeau, "On the uniform sampling of cielab color space and the number of discernible colors," in *4th International Conference on Computational Color Imaging*, 2013.

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

[62] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2006.

[63] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proceedings of the Sixth International Conference on Computer Vision*, 1998.

[64] J. F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[65] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *International Conference on Computer Vision and Pattern Recognition*, 2005.

[66] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1989.

[67] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization – artificial ants as a computational intelligence technique," *IEEE Computational Intelligence Magazine*, vol. 1, pp. 28–39, 2006.

[68] I. Gallo, M. Vanetti, S. Albertini, and A. Nodari, "Multi-net system configuration for visual object segmentation by error backpropagation," in *6th Iberian Conference on Pattern Recognition and Image Analysis*, 2013.

[69] I. Gallo, A. Zamberletti, S. Albertini, and L. Noce, "High entropy ensembles for holistic figure-ground segmentation," in *Proceedings of the British Machine Vision Conference*, 2014.

[70] P. Dollár and L. Zitnick, "Structured forests for fast edge detection," in *International Conference on Computer Vision*, 2013.

[71] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[72] E. Borenstein, E. Sharon, and S. Ullman, "Combining top-down and bottom-up segmentation," in *International Conference on Computer Vision and Pattern Recognition*, 2004.

[73] P. Dollár, P. Welinder, and P. Perona, "Cascaded pose regression," in *International Conference on Computer Vision and Pattern Recognition*, 2010.

[74] M. E. Nilsback and A. Zisserman, "Delving deeper into the whorl of flower segmentation," *Image and Vision Computing*, vol. 28, no. 6, pp. 1049–1062, 2010.

[75] M. Marszałek and C. Schmid, "Accurate object localization with shape masks," in *International Conference on Computer Vision and Pattern Recognition*, 2007.

[76] L. Bertelli, T. Yu, D. Vu, and B. Gokturk, "Kernelized structural SVM learning for supervised object segmentation," in *International Conference on Computer Vision and Pattern Recognition*, 2011.

[77] M. Seyedhosseini, M. Sajjadi, , and T. Tasdizen, "Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks," in *International Conference on Computer Vision*, 2013.

[78] Y. Chai, V. Lempitsky, and A. Zisserman, "BiCos: A bi-level co-segmentation method for image classification," in *European Conference on Computer Vision*, 2011.

[79] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *International Conference on Computer Vision*, 2009.

[80] G. Guennebaud, B. Jacob *et al.*, "Eigen c++ library," http://eigen.tuxfamily.org/, 2010.