Multi-net System Configuration for Visual Object Segmentation by Error Backpropagation

Ignazio Gallo, Marco Vanetti, Simone Albertini, and Angelo Nodari

Università dell'Insubria, Dipartimento di Scienze Teoriche e Applicate
via Mazzini 5, 21100 Varese, Italy
ignazio.gallo@uninsubria.it

Abstract. This work proposes a new error backpropagation approach as a systematic way to configure and train the Multi-net System MNOD, a recently proposed algorithm able to segment a class of visual objects from real images. First, a single node of the MNOD is configured in order to best resolve the visual object segmentation problem using the best combination of parameters and features. The problem is then how to add new nodes in order to improve accuracy and avoid overfitting situations. In this scenario, the proposed approach employs backpropagation of error maps to add new nodes with the aim of increasing the overall segmentation performance. Experiments conducted on a standard dataset of real images show that our configuration method, using only simple edges and colors descriptors, leads to configurations that produced comparable results in visual objects segmentation.

Keywords: multiple neural networks configuration, visual object segmentation

1 Introduction

Visual object segmentation is still an open problem and represents an important and difficult task in computer vision and image analysis. In this work, our focus is on the configuration of a recent work [7] which proposes a multi net model that combines multiple sliding-window based descriptors in order to provide a generic segmentation framework for visual objects. The aforementioned method, known as "Multi-Net for Object Detection" (MNOD), is highly versatile because it may combine very effective local descriptors, i.e. those proposed by [15, 4], and simple features with a hierarchical supervised learning process which allows to distribute the image segmentation task on multiple units, each of which is focused on a particular image characteristic at a particular scale. The MNOD we use in this work address the problem of performing class-specific object segmentation, i.e., automatic segmentation using annotated training images. Its goal is to segment an image into regions, where each region solely contains one or more objects which belong to a fixed class. As object segmentation requires that each segmented region be a semantic object, it is much more challenging than traditional image segmentation [11], which only requires that each region is a homogeneous texture, but at the same time it is less challenging of the multi-class object segmentation as approached in [3]. Although MNOD offers great versatility and the configuration proposed in the original paper leads to good segmentation results for

many object classes, it does not provide a general strategy to configure its wide parameter space. In particular, the set of free parameters includes the tree topology and, for each node, the set of features extracted from the input image, the size of the sliding window and the downscaling ratio.

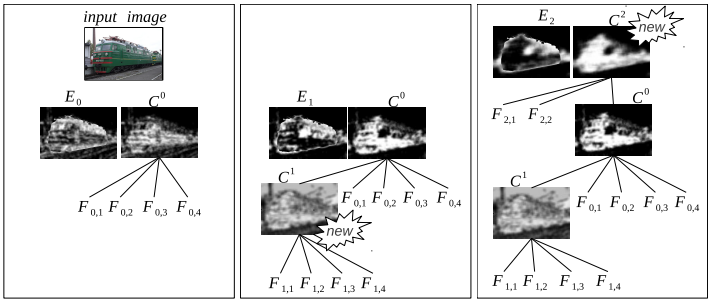
In literature several strategies have been proposed, using simple greedy algorithms [10], boosting strategies [15], genetic algorithms [12] and ant colony optimization [1], while other methods have been proposed to automatically configure the architecture of backpropagation networks [2, 9]. In this work we propose and evaluate a novel approach to automatically configure the MNOD model by performing at the same time a feature selection and a network architecture organization. We propose an incremental learning algorithm for the network structure through an optimized search. Our solution uses a greedy feature selection and an error backpropagation strategy to automatically add new nodes to the MNOD structure, selecting the parameters that better reduce the segmentation errors. The process starts from a single node and tries to configure it in the best way in order to maximize the training accuracy. Error maps are computed in order to add a new node that best solves the unsolved errors. This process is repeated until no improvement is obtained, in a deep learning style. It reminds the typical boosting framework, which uses the error map to weight the error of several weak learners which combined together lead to a strong learner [13]. For this reason we compare our method against the AdaBoost algorithm [6].

2 The MNOD Model

The basic idea of the MNOD is to use many neural models trained individually to resolve a part of the object segmentation problem. Here, we summarize briefly the Multi-Net for Object Detection (MNOD) model as proposed in [7]. The MNOD is a Multi-Net System which consists of a tree of neural networks [14]. Each node n, properly configured with its own parameters $\mathbf{P}$, acts like an independent supervised neural network $C^n_{\mathbf{P}}$. The task of each neural network is to refine and optimize the segmentation map provided by its children. The output of each node is a segmentation map which can be considered as a two-dimensional map describing the likelihood that each pixel of the image belongs to the object of interest, while the inputs are the output of its children. This model allows the diversification of a node $C^n_{\mathbf{P}}$ by adjusting the parameters $\mathbf{P}$ and selecting the appropriate leaves and internal nodes. We can refer to $\mathbf{P}_n = \{I_s, W_s\}$ as the configuration for the node n, given I_s and W_s respectively image size and sliding window (or kernel) size. Just using different combinations of these two parameters, the model becomes specialized to segment specific objects of a given class.

3 The Proposed Configuration Method

The task of finding an optimal configuration of the MNOD model is complex due to the very large parameter space involved in the search. In this work we propose an algorithm that, starting from a configuration with a single node, iteratively tries to add new nodes able to deal with the segmentation problem for particular locations of the image.



(a) (b) (c)

Fig. 1: An example of how the configuration algorithm builds a simple MNOD. In (a), the C^0 node configured by the algorithm and the segmentation map obtained from the *input image*. In (b), the first node C^1 built using the error map E_0. In (c), the new layer with C^2 as the root node.

We introduce the concept of errors map E_{ij}, built for each training image I_j and computed for a single node C^i. A single E_{ij} is computed for each pixel such that $E_{ij}(x,y) = |O_j(x,y) - \overline{O}_{ij}(x,y)|$, where O_j is the expected segmentation map for the training image I_j and $\overline{O}_{ij}$ is the output segmentation map obtained from the node C^i of the MNOD tree on the same image. The map E_{ij} is used to train a new node on a particular set of problematic points that the node C^i was not able to solve. Setting this new node as a child of C^i, the new model may increase its ability to recognize the objects of interest. In this case the node is added to the tree, otherwise it will be dropped. Every time a new node is added, if we want to add further nodes, the root node is trained again and all the maps E_{ij} are recalculated. Given E_{ij} we obtain a set of points $\mathbb{P} = \{(x,y)|E_{ij}(x,y) > t\}$ randomly extracted considering only the points with error greater than the threshold t. Fig. 1 shows an example of how the maps of errors E_0 and E_1 evolve as a new node is added to the tree. The same picture is an example which reveals that dark pixels (with low error value) increase by adding a child node (trained on the problematic pixels contained in E_0) to the first node, or by adding a new layer having a single root node C^2. The map E_1, computed after the introduction of the node C^1, as well as the map E_2 computed after the introduction of C^2 shows how the segmentation outcome improves. Note that E_1 and E_2 have been included in the figure in order to highlight the improvements introduced by the added nodes but they are not directly involved in the process.

The basic idea is to configure the model incrementally, trying to improve the segmentation results whenever new nodes are added. The overall process starts from a single node configured with the best parameters and leaf nodes (see the Algorithm 1 when E and C^{t-1} are empty). So iteratively we try to add new configured nodes as children of the root node as long as it helps to increase the segmentation accuracy (see the **repeat/until** loop of Algorithm 2). Then, the iteration is repeated by adding a new layer with a new root node which receives in input the segmentation map of the previously

configured layer (see the **for** loop of Algorithm 3). The proposed *BPConf* algorithm is composed of two main parts: the first tries to configure a single node in the best way (see examples in Fig. 1a and 1c), while the second tries to improve the accuracy of a single node backpropagating the errors to a new input node (see example in Fig. 1b). The algorithm $GenerateNode$ (see Algorithm 1) is recursive, that is it tries to add a new sub-node to the current node until it reaches the level $t = 0$ or until the accuracy calculated on the starting node does not degrade.

In the following section, using a very large and difficult dataset of real images, we show why the *BPConf* algorithm is able to configure the MNOD model.

Algorithm 1 *BPConf*: $GenerateNode$ to configure a node of the level t.
$child(T, N)$ returns the root node of a new tree obtained connecting the root node T of a new tree as child of the the root node N of an existing sub-tree.

Require: C^{t-1}, eventual previous level root node of a sub-tree,
E, eventual error map for training the sub-tree,
$\mathbb{I}_t = \{I_t^1, I_t^2, \ldots, I_t^{N_i}\}$ space of image sizes for a node,
$\mathbb{W}_t = \{W_t^1, W_t^2, \ldots, W_t^{N_w}\}$ space of sliding windows sizes for a node,
$\mathbb{F}_t = \{F^1_{P_1,t}, F^2_{P_2,t}, \ldots, F^{N_f}_{P_{N_f},t}\}$ set of possible leaf nodes with configurations P_i.

if $E = \perp$ **then** $E \leftarrow$ map of random selected points
$C^t \leftarrow \perp$
$accuracy \leftarrow 0$ {defined on equation (1)}
for all $\mathbf{P} = \{I^k, W^m\} \in \mathbb{I}_t \times \mathbb{W}_t$ **do**
  $C \leftarrow child(C^{t-1}, C_{\mathbf{P}})$
  for all $F \in \mathbb{F}_t$ **do**
    $\overline{C} \leftarrow child(F, C)$
    $result \leftarrow$ segmentation accuracy of $\overline{C}$ evaluated on validation set after train using E
    if $result$ is greater than the previous **then**
      $chosenNode \leftarrow \overline{C}$
      $chosenAccuracy \leftarrow result$
    end if
  end for
  if $chosenAccuracy > accuracy$ **then**
    $accuracy \leftarrow chosenAccuracy$
    $C^t \leftarrow chosenNode$
  end if
end for
for all $F \in \mathbb{F}_t$ **do**
  $C \leftarrow child(F, C^t)$
  $result \leftarrow$ train using E and evaluate training accuracy for C
  if $result > accuracy$ **then** $accuracy \leftarrow result$ and $C^t \leftarrow C$
end for
return C^t

4 Experiments

In this study we applied our configuration method to the MNOD model in order to compare its results with the results achieved by all the methods that have participated to 'The PASCAL Visual Object Classes Challenge" [5]. A second experiment we conducted is the comparison of the *BPConf* with the standard configuration method *AdaBoost*.

Algorithm 2 *BPConf*: $GenerateLayer$ algorithm

Require: t is the current layer of the tree,
C^{t-1} the root node of the previous layer.

$C^t \leftarrow GenerateNode(\perp, C^{t-1})$
repeat
  $E \leftarrow$ error map of C^t
  $node \leftarrow GenerateNode(E, \perp)$ {parameters: error map, sub-tree}
  $C \leftarrow$ add $node$ as child to a clone of C^t
  $result \leftarrow$ train and evaluate segmentation accuracy on C on random selected points map
  if $result > accuracy$ {initialized in algorithm 1} **then**
    $C^t \leftarrow C$ {It keeps the added simple tree}
  end if
until No node is added
return C^t

Algorithm 3 *BPConf*: $GenerateTree$ algorithm

Require: T, the max number of layers

$C^0 \leftarrow \perp$ {no sub-tree for the first layer}
for $t \in \{1, \ldots, T\}$ **do**
  $C^t \leftarrow GenerateLayer(C^{t-1})$
end for
return C^T

We made the experiments using the VOC2011 [5], a standard and very challenging dataset composed of 20 classes of objects and 5.034 manually annotated segmentations divided into train and validation sets. The use of a multi-class dataset for testing a binary segmentation algorithm is, in our opinion, a reasonable choice considering we want to validate the effectiveness of the configuration method which would lead to reasonable results rather than confronting the MNOD algorithm against other binary segmentation algorithms.

For these experiment we used the segmentation accuracy index defined in equation (1), proposed by the contest [5] and usually called *Jaccard index* [8].

$$Acc = \frac{TP}{TP + FP + FN} \tag{1}$$

The values under consideration are calculated pixel-by-pixel: TP are the True Positives, FP the False Positives and FN the False Negatives.

4.1 Comparison using a standard dataset of real images

The main goal of this experiment is to demonstrate that the *BPConf* algorithm produces good MNOD configurations, comparing the class-specific object segmentation results obtained with the the best multi-class object segmentation results published in the competition VOC2011 [5].

In this experiment for simplicity we set the maximum number of layers T to three and for each of these layers, we defined a range of variability on the following parameters: (i) type and scale ratio of leaf nodes, (ii) size of the input image I_s, (iii) size of the sliding window W_s, (iv) maximum number N_{max} of children nodes in input to each internal node. Note that adding more layers we obtain results very slightly better, but

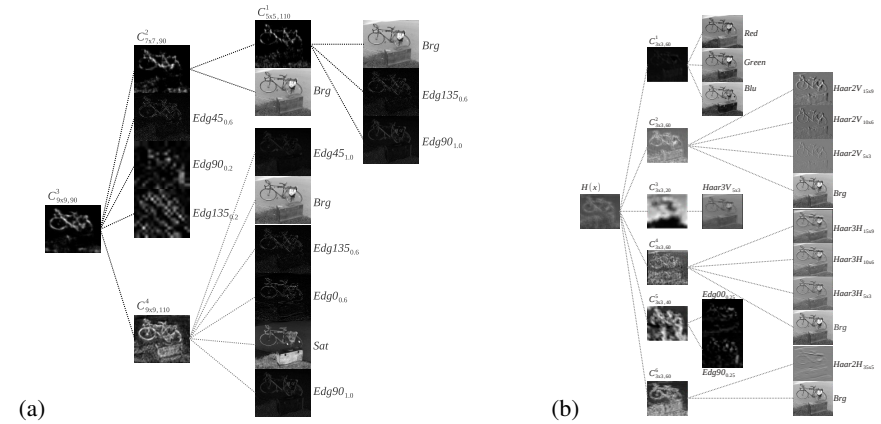


Fig. 2: MNOD tree example created by the *BPConf* algorithm (a) and AdaBoost algorithm (b) for the class "bicycle" of the VOC2011 dataset. In (b) the $H(x)$ node is the output of the AdaBoost algorithm.

the complexity and training time greatly increases. The set of leaf nodes $\mathbb{F}_t$ contains four simple leaf nodes, which extract high-frequency information from four different edge orientations: $0°$, $45°$, $90°$ and $135°$, plus the three channels *brightness (Brg)*, *Hue* and *saturation (Sat)*. Before extracting these edge features, each leaf node performs an image downscaling (driven by the parameter I_s) using the following scale factors: $\{0.2, 0.6, 1.0\}$. We use the same type of leaf nodes and scalings for all the three layers of the tree and we set $N_{max} = 6$ for all the nodes. The three different sets of image sizes $\mathbb{I}_0 = \{90, 100, 130\}$, $\mathbb{I}_1 = \{50, 70, 90\}$ and $\mathbb{I}_2 = \{20, 50, 80\}$, were used to configure the nodes of the three layers respectively, while the same set of sliding windows sizes $\mathbb{W}_t = \{2 \times 2, 5 \times 5, 9 \times 9\}$ was used to configure all the nodes.

The *BPConf* column of Table 1 shows the results obtained applying the *BPConf* algorithm in order to generate a MNOD tree for each class of objects of the VOC2011 dataset. The values in this table represent the accuracy results on object segmentation, when the metric explained by equation (1) is used.

Table 1 compares the accuracy results of an MNOD configured with the *BPConf* algorithm against the results obtained with the *AdaBoost* configuration algorithm. These two results are compared with the best results of the VOC2011 contest. The numerical results show that our configuration method is valid because using the same initialization for all the classes we obtained many results not so far from the best values achieved in the competition VOC2011 [5] and in many cases we got better results. We remark that our algorithm is a single class segmentator and do not perform any kind of multiclass distinctions. We built with *BPConf* a specific model for each different class in the dataset. Therefore we can conclude that with our configuration algorithm we can obtain good results for binary segmentations, considering it gives results similar to the best obtained in the competition thus outperforming some of them.

The Fig. 2a shows an example of how the proposed *BPConf* algorithm tries to improve the segmentation results by adding new nodes to a MNOD model, through the

Table 1: Comparison between the Jaccard index measure obtained with the MNOD when configured with the proposed *BPConf* algorithm, and all the best results obtained by the participants of the Pascal segmentation challenge using the *VOC2011* dataset.

Class name	VOC2011	*BPConf*	*AdaBoost*	Class name	VOC2011	*BPConf*	*AdaBoost*
aeroplane	54.30	**56.46**	13.28	diningtable	30.1	**30.17**	15.32
bicycle	23.90	**23.90**	4.53	dog	33.90	**42.35**	21.05
bird	**46.00**	27.57	12.00	horse	49.10	**52.23**	20.34
boat	35.30	**40.85**	12.14	motorbike	54.4	**57.12**	22.77
bottle	**49.40**	26.97	17.49	person	**46.4**	35.76	16.88
bus	**66.20**	64.53	33.93	pottedplant	28.8	**35.31**	10.64
car	**56.20**	41.64	20.07	sheep	51.3	**55.50**	20.94
cat	46.10	**46.29**	25.63	sofa	26.40	**35.87**	22.58
chair	15.00	**15.02**	11.89	train	44.9	**56.56**	23.35
cow	47.4	**55.86**	23.23	tvmonitor	**45.8**	30.53	18.50
Mean	42.55	41.52	18.32				

mechanism of the error maps backpropagation. The model was trained to segment objects belonging to the class *bicycle* of the VOC2011 dataset. From this figure it is evident that the node C^4, constructed with the *BPConf* algorithm, contributes to improve the final segmentation map produced by the C^3 node.

4.2 Comparison with the AdaBoost algorithm

With the aim to compare the proposed method with a standard configuration method, we have implemented the well known AdaBoost algorithm [6].

AdaBoost selects a MNOD's node C^h from the family of weak classifiers $\mathbb{H}$ after each cycle $h = 1, \ldots, H$. We made several tests using different values for the parameter H that identifies the maximum number of weak learners to select for a configuration of a MNOD, and at the end we chose the value $H = 6$ which gave the best result. $\mathbb{H}$ contains all the individual nodes C^i constructed using all the combinations of the same sets of parameters $\mathbb{I}_t$ and $\mathbb{W}_t$ of the previous experiment in section 4.1, while the set of features $\mathbb{F}_t$ of the previous experiment was extended with some more powerful $Haar$ [15] features and the Red, $Green$ and $Blue$ channel. We extended the number of features in order to obtain a slightly more interesting result from the MNOD configured with the AdaBoost algorithm. The *AdaBoost* column in Table 1 shows the results obtained applying the *AdaBoost* algorithm in order to generate a MNOD tree for each class of objects of the VOC2011 dataset. The values in this table represent the accuracy results on object segmentation, when the metric explained by equation (1) is used. The numerical results confirm that our *BPConf* configuration method is very interesting compared with the AdaBoost results.

5 Conclusions

The method proposed in this work makes a significant contribution to the MNOD segmentation approach presented for the first time in [7]. In fact, MNOD is a very powerful segmentation method, but does not have a strategy for the automatic configuration of its parameters, features and topology. Our proposed configuration idea is very simple and suitable for models with tree structures such as the MNOD. Analyzing the results obtained in the VOC2011 competition and comparing them with those obtained in this work using our configuration method and the AdaBoost method, it emerged that the proposed *BPConf* algorithm allows to obtain very good results for many classes. If we also consider that for the experiments conducted in this paper we used only very simple features, we may conclude that the possibility to further improvements exists.

References

1. Ahmed Al-ani. Ant colony optimization for feature subset selection. In *Enformatika Conferences*, pages 35–38, 2005.
2. Timur Ash. Dynamic node creation in backpropagation networks. *Connection Science*, 1:365–375, 1989.
3. J. Carreira and C. Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *IEEE International Conference on Computer Vision and Pattern Recognition*, June 2010. description of our winning PASCAL VOC 2009 and 2010 segmentation entry.
4. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, pages 886–893, 2005.
5. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge (VOC) 2007-2011 Results. http://www.pascal-network.org/challenges/VOC.
6. 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*, 55(1):119 – 139, 1997.
7. Ignazio Gallo and Angelo Nodari. Learning object detection using multiple neural netwoks. In *VISAPP 2011*. INSTICC Press, 2011.
8. Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, February 1912.
9. Jiqian Liu and Yunde Jia. *Dynamic Construction of Multilayer Neural Networks for Classification*. 2011.
10. Xiuwen Liu and Deliang L. Wang. Texture classification using spectral histograms. *IEEE Transactions on Image Processing*, 12:661–670, 2003.
11. Yi Ma, Harm Derksen, Wei Hong, and John Wright. Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1546–1562, 2007.
12. Diego S. C. Nascimento, Anne M. P. Canuto, Ligia M. M. Silva, and Andre L. V. Coelho. Combining different ways to generate diversity in bagging models: An evolutionary approach. In *International Symposium on Neural Networks*, pages 2235–2242, 2011.
13. Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, jul 1990.
14. Amanda J.C. Sharkey. *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, chapter Multi-Net Systems. Springer, 1999.
15. Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2001.