Deliverable D6.3

Description and evaluation of techniques for transfer learning across sub-categories

Contract number: **FP7–216529** PinView

Personal Information Navigator Adapting Through Viewing

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under *grant agreement* n° 216529.

Identification sheet

Project ref. no.	**FP7–216529**
Project acronym	PinView
Status and version	Final, Revision: 1.0
Contractual date of delivery	30.09.2009
Actual date of delivery	13.10.2009
Deliverable number	D6.3
Deliverable title	Description and evaluation of techniques for transfer learning across sub-categories
Nature	report
Dissemination level	PU – Public
WP contributing to the deliverable	WP6 Semantic sub-categorisation features
Task contributing to the deliverable	Task 6.3 Sub-categorisation with limited data
WP responsible	Xerox Research Centre Europe
Task responsible	Xerox Research Centre Europe
Editor	Teófilo E. de Campos, <t.de-campos@xrce.xerox.com>
Editor address	XRCE, 6, chemin de Maupertuis, 38240 Meylan, France
Authors in alphabetical order	Martin Antenreiter, Peter Auer, Gabriela Csurka, Teófilo E. de Campos, Zakria Hussain, Jorma Laaksonen, Julian McAuley, Ronald Ortner, Kitsuchart Pasupa, Florent Perronnin, Craig Saunders, John Shawe-Taylor, Ville Viitaniemi
EC Project Officer	Pierre-Paul Sondag
Keywords	Meta classifiers, structured learning, image classification, image region labeling.
Abstract	This report presents contributions in sub-categorisation. In the first part we propose a simple approach to build a meta-learner on individual classifiers that is supposed to implicitely learn class inter-dependencies. We further study how the performance is modified if we add automatically learned meta-classes (group of similar classes). In the second part, we address the problem using a structured learning approach. The main idea is to build a graphical model which efficiently performs labeling at multiple levels simultaneously.

List of annexes

none

Contents

1 Overview

This is a Deliverable of the *Personal Information Navigator Adapting Through Viewing*, PinView, project, funded by the European Community's Seventh Framework Programme under Grant Agreement n° 216529.

The report constitutes the last output of the Work Package 6 *Semantic sub-categorisation features* which aims at improving the state-of-the-art in visual categorisation performance. The main research outcomes of Task 6.3 *Sub-categorisation with limited data* are described here.

According to the description of work (DoW), task 6.3 of PinView *concerns the limited data available for sub-categorisation. Tautologically, sub-categories are associated with fewer data than categories, and rare sub-categories with fewer still.* To overcome the lack of training data for a given category, a natural solution is to make use of data from other categories, especially from related ones. This requires some sort of joint learning of sub-categories. Hence, we first propose hierarchies of classes to grasp the inter-dependencies that the individual learners may not have properly captured. We will show however that, although the learned semantic hierarchies are very reasonable from the perspective of a human, they do not lead to an increase in classification accuracy and may even incur a loss. This part of the work will be continued in WP 5.3 (feature selection) and the results will be delivered in the corresponding deliverable.

The DoW of Task 6.3 of PinView also points that parts and sub-parts of objects (and images) may have very different mean appearances. This indicates that there is a need for a method which learns the relationship between parts and sub-parts of objects. Hence, in a second part, we propose to learn a model of parts hierachies. We will show that it can lead to a significant increase in image region labeling accuracy.

2 Introduction

The art of multilabel classification lies in reliably detecting and exploiting interdependencies between classes. A base classifier that is trained on enough examples will also learn these interdependencies. However, in the common case where the training set is not sufficiently large, a base learner e.g. for cars may not fully grasp the interdependency between cars and roads. This defect can be corrected by having a base learner for roads that will be able to learn roads from more training examples than just those where also cars appear. That way, combining the road classifier with the car classifier in a suitable way will also improve the performance for classification of cars.

Following this principle, in the section 3 we present a simple method for multilabel learning based on the combination of binary base classifiers. The main idea is to train binary classifiers and then feed the confidence scores of these base learners into a support vector machine (SVM) in order to improve prediction accuracy. The idea behind this is that in contrast to stacking approaches where the combination of base learners is learnt explicitely, here the meta-level learner grasps also inter-dependencies that the base learners have not properly captured.

The DoW of Task 6.3 of PinView also points that parts and sub-parts of objects (and images) may have very different mean appearances. This indicates that there is a need for a method which learns the relation between parts and sub-parts. In the second part of this deliverable 4, we propose therefore a method for image and sub-image categorisation which can benefit from a hierarchy of categories that is analogous to the notion of *inheritance* in object-oriented programming. More general categories are more likely to occupy larger portions of an image or even the whole image, whereas more specific sub-categories are topologically contained in categories. We encoded this type of relationship in a graphical model in which sub-parts always correspond to lower levels than parts. We employed structured learning to learn the relationship between the appearance of parts and sub-parts (even though we may not not necessarily assign different labels to parts/sub-parts). This enables a structured description of images which can be used to label its regions, an essential step towards *semantic* image segmentation.

3 Learning Meta Classifiers and Meta Classes

In multilabel classification problems there are usually interdependencies between classes. For example, when labeling images it is rather unlikely that an image containing a sheep also shows an aeroplane. On the other hand, images containing cars will usually also contain roads. The art of multilabel classification lies in reliably detecting and exploiting these interdependencies. A base classifier that is trained on enough examples will also learn these interdependencies. However, in the common case where the training set is not sufficiently large, a base learner for cars may not fully grasp the interdependency between cars and roads. This defect can be corrected by having a base learner for roads that will be able to learn roads from more training examples than just those where also cars appear. That way, combining the road classifier with the car classifier in a suitable way will also improve the performance for classification of cars.

In this section, we present a simple method for multilabel learning based on the combination of binary base classifiers. That is, we propose to train for each label a binary base classifier. Then we feed the output (i.e., the confidence scores) of these binary base learners into a support vector machine (SVM) in order to improve prediction accuracy.

The basic concept underlying this simple approach resembles *stacking* [45] methods: In the stacking framework the output of several (distinct) base classifiers is combined by a meta-level learner to give an improvement in (binary or multiclass) classification. In the multiclass case usually multiclass classifiers are used as base learners and as meta-level learners (cf. [8, 34]).

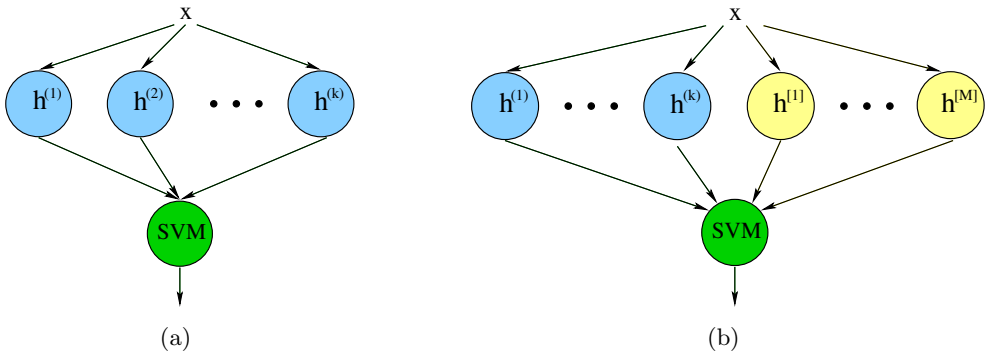


Figure 1: Fig. 1(a) shows the two layer architecture for learning the interdependencies between the individual classes and Fig. 1(b) the two layer architecture with additional meta-classes classifiers (marked yellow).

While the idea behind stacking is that the metalearner shall be able to combine the base learners in a more sophisticated way than doing simple voting or cross-validation [45], in our method the meta-level learner shall grasp interdependencies between the single classes that the base learners have not properly captured.

3.1 Learning Meta Classifiers

Our algorithm uses binary base classifiers $h^{(k)}(x)$ that are trained to recognize single classes C_k. In order to better capture interdependencies between the individual classes we propose combining the confidence scores of the different base classifiers by simply feeding them into a support vector machine (SVM). The two layer approach is shown in Fig. 1(a). We consider the following *multilabel* problem: Given is a set of training examples $\{x_1, \ldots, x_n\} \subset X$ together with labels for N different classes $\{C_1, \ldots, C_N\}$ (where each $C_k \subseteq X$). That is, for each training instance x_i and each class C_k the respective label is

$$y_i^{(k)} := \begin{cases} +1 & \text{if } x_i \in C_k \\ -1 & \text{otherwise.} \end{cases}$$

As the problem is assumed to be *multilabel* but not necessarily *multiclass*, $y_i^{(k)} = 1$ not necessarily implies that $y_i^{(\ell)} = 0$ for $\ell \neq k$.

We first train for each class C_k a corresponding (binary) base learner $h^{(k)}$ that shall be able to predict the labels $y_i^{(k)}$ well. Each base classifier returns a confidence score $s_i^{(k)}(x_i)$ for each training example x_i. In our case this will be a real value (the distance to the separating hyperplane) that is positive if the classifier predicts that x_i is in the target class and negative otherwise. Let $s^{(k)}(x)$ be the confidence score returned by base classifier $h^{(k)}$ for class C_k on training instance x. The collected confidence scores are then used by N metalearners, one for each class C_k, where the training set consists of the vectors

$$\mathbf{v}_i = \left(s^{(1)}(x_i), s^{(2)}(x_i), \ldots, s^{(N-1)}(x_i), s^{(N)}(x_i)\right) \tag{1}$$

for all training instances x_i. The label of each $\mathbf{v}_i$ is simply the label $y_i^{(k)}$ of x_i with respect to class C_k.

3.2 Adding Meta-classes based learners

The idea of our second method is to use information of the meta-classes (group of classes) to improve the performance of our two layer approach described in 3.1. This is done as

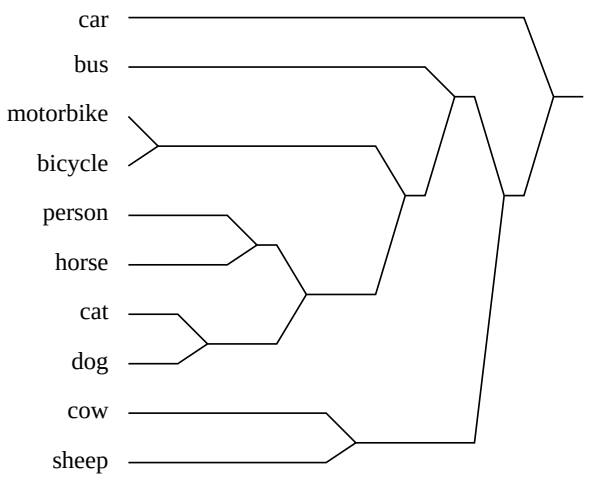


Figure 2: Taxonomy tree of visual similarity of the classes on the VOC2006 dataset.

follows. In addition of the N base learners $h^{(k)}$, we train in addition M binary classifiers $h^{[1]}(x_i), \ldots, h^{[M]}(x_i)$ one for each meta-class. The confidence scores from the N base learners and the M meta-class base learners are concatenated to a new vector:

$$\mathbf{v}_i = \big(s^{(1)}(x_i), \ldots, s^{(N)}(x_i), s^{[1]}(x_i), \ldots, s^{[M]}(x_i)\big). \tag{2}$$

This vector is used for learning the SVM in the second stage (see Fig. 1(b)).

In which follows, we show that visual similarities between the classes can be used in data-driven creation of a taxonomy of object classes and allow automatically building a meta-class structure based on the similarity between classes.

3.3 Automated Construction of Meta-classes

Here we present our method to build a meta-class structure based on the similarity of classes. To do this, first, we train for each class C_k a corresponding base learner $h^{(k)}$ and then analyze the performance of the binary base learners. The main idea is to group classes which are difficult to separate in a so-called "new meta-class".

For that purpose, we first calculate pairwise misclassification rates between all class pairs. That is, we assessed the similarity of two classes by studying the difficulty of separating them. For achieving optimal classification we used a k-nearest neighbor classifier with optimization of the k value and sequential forward selection (SFS) of the used visual features. When the two classes which were the most difficult to separate were found, we created a virtual joint class of those two and re-calculated all pairwise misclassification rates. We repeated the joining process until all classes were grouped.

Meta-classes on the VOC2006 Dataset. As an example, we show how this works in the case of Pascal Visual Object Classes Challenge 2006 (VOC2006) dataset [12] (see details in 3.4). For the experiments, we used the low-level features of the PicSOM image analysis framework [24]. The joining process produced the unbalanced binary tree of the ten classes shown in Figure 2. We argue that even though this visual class taxonomy has been created purely from low-level visual feature data, it coincides to some extent with the corresponding semantic class taxonomy one might come to by mental contemplation.

In the VOC data, there exist class pairs (*motorbike*, *bicycle*), (*cat*,*dog*) and (*cow*,*sheep*) which tend to be misclassified and are therefore combined in the early stages of the taxonomy tree formation process of Figure 2. Some of the later merges are, however, not as straight-

forward to interpret. (One may actually state that such an iterative merging process which only relies on local information is bound to be chaotic.)

The same principle can be used also to create a one-dimensional, non-hierarchical ordering of the classes where mutually similar ones reside near each other. In order to achieve this, we again calculated the misclassification rates between all class pairs and regarded the misclassification rate between each class and itself as 100%. For each class, all the classes were then ordered in the order of descending misclassification rate. In this process, the ordinal number 0 was always assigned to the class itself, number 1 to the class with the highest misclassification rate, ie. the highest visual similarity, number 2 for the second highest misclassification rate, etc. This led to a square matrix where each row was associated with one class and the numbers in the columns (ordered in the same order as the rows) displayed the similarity rankings of that class with all classes. We then swapped the rows and columns of the matrix so that the smallest numbers we forced to reside along the diagonal and the first subdiagonals whereas the largest numbers were moved to the anti-diagonal corners. The resulting ordering and *ordered similarity matrix* is shown in Table 1.

bus	0	3	2	4	1	5	6	7	9	8
car	3	0	2	4	1	5	6	7	9	8
motorbike	4	5	0	1	2	3	6	7	9	8
bicycle	7	6	1	0	2	4	5	3	8	9
person	7	6	1	3	0	2	5	4	9	8
horse	9	8	3	6	1	0	3	2	5	7
cat	8	9	5	4	3	2	0	1	5	7
dog	8	9	7	6	3	2	1	0	5	4
cow	8	8	7	6	5	2	4	3	0	1
sheep	9	8	5	7	4	3	6	2	1	0

Table 1: Ordered similarity matrix of the VOC2006 object classes.

It can be seen again that there exist obvious pairs (*motorbike*, *bicycle*), (*cat*,*dog*) and (*cow*,*sheep*), but they are now aligned on a linear continuum between the two opposites, *bus* and *sheep*.

Meta-classes on the VOC2009 Dataset. After the inspection of the visual taxonomy tree for the VOC2006 dataset, we built the tree for the Pascal Visual Object Classes Challenge 2009 (VOC2009) dataset [11] (see details in 3.4). The tree structure for VOC2009 dataset is shown in Fig. 3.

It can be seen that the classes *car*, *bus*, and *train* are difficult to separate. Additionally, the classes *bicycle* and *motorbike* are also difficult to classify and are grouped as a new meta-class *two-wheels*. The classes *aeroplane*, *boat*, and *tv/monitor* are at the top of the tree, which means they are not similar to other classes. The generated grouping looks plausible and therefore we can assign meaningful class names for the meta-classes.

The tree for VOC2009 has $M = 8$ meta-classes (yellow ellipses), which are used to train additional binary base learners $h^{[1]}(x_i), \ldots, h^{[M]}(x_i)$ in Equation (2).

3.4 Experiments

We conducted experiments on the image classification database taken from the Pascal Visual Object Classes Challenges 2006 (VOC 2006) [12] and 2009 (VOC2009) [11]. The VOC 2006 dataset contains 10 classes in 5,304 images, on which a total of 9,507 annotated objects can be found. The VOC 2009 dataset contains 20 classes in 9,963 images with 24,640 annotated

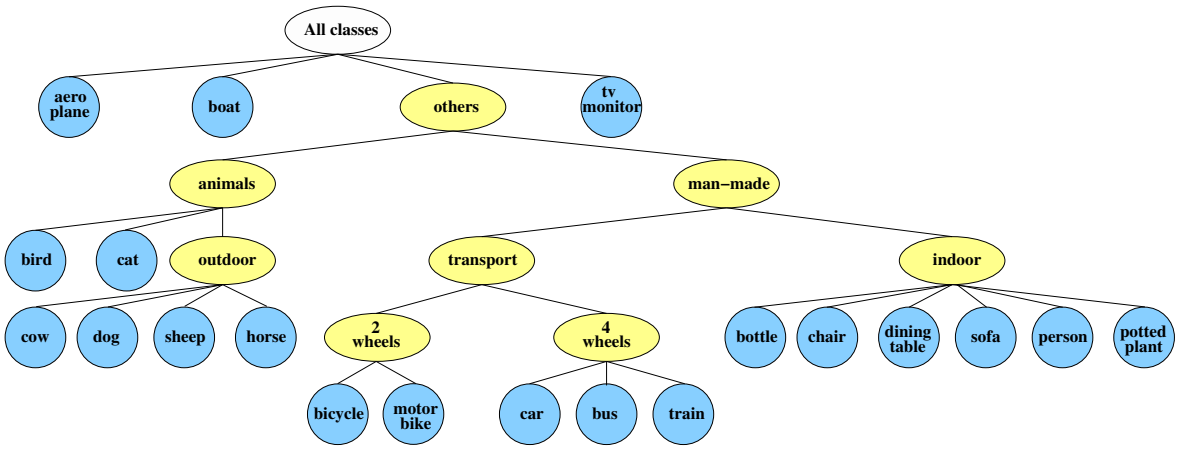


Figure 3: VOC2009 tree structure generated by merging similar object classes based on the similarity matrix. The yellow ellipses are new meta-classes.

objects. Both are multilabel datasets. These datasets are split into a fixed training, validation, and test set. In the VOC 2006 database the training set contains 1,277 images. The validation set has 1,341 images and the test set 2,686 images. The VOC2009 database has 3,473 training images, 3,581 validation images, and 6,650 test images. From the datasets we extracted various visual features for learning. A list of the extracted features can be found in Tab. 2.

f_1 ... texture statistics of segments [15]
f_2 ... subsampled grayvalues
f_3 ... subsampled grayvalues (intensity normalized)
f_4 ... basic moments
f_5 ... basic moments (intensity normalized)
f_6 ... moment invariants [16]
f_7 ... moment invariants (intensity normalized)
f_8 ... SIFTs [29]
f_9 ... PCA-SIFTs [20]
f_{10} ... Color-SIFTs [43]

Table 2: Feature types for the experiments

The training of the base learners was done on the training dataset using the independent validation set for parameter selection. For learning the second stage of the architecture, the selection of the kernel and the parameters was done using 5-fold cross-validation on the validation data. After that, we learned the combined classifiers using the validation data. All reported results are from the evaluation on the test data.

Classification Experiments without using Meta-class Information. We compare our results with a simple one-vs-all classifier architecture. Each one-vs-all classifier was trained using the training and validation set together and the parameter selection was done with using 5-fold cross-validation. As evaluation criterion we use the the average precision (AP).

All results for the VOC2006 can be found in Tab. 3. We used for that experiment a linear kernel for the SVM. The original one-vs-all classifier is outperformed by our algorithm by

9.53%.

class	original one-vs-all	combination linear	impr.
bicycle	61.12	61.36	0.24
bus	27.32	51.66	24.34
car	70.92	74.03	3.11
cat	24.41	37.13	12.72
cow	18.92	23.41	4.49
dog	25.88	32.16	6.28
horse	12.12	22.44	10.32
motorbike	33.19	47.09	13.90
person	35.16	42.55	7.39
sheep	29.39	41.87	12.48
avg	33.84	43.37	9.53

Table 3: Comparison of the classifier using a one-vs-all architecture, and our approach with a linear kernel on the VOC 2006 dataset. Results are in percentage using the average precision measure.

For the VOC2009 we used a linear, polynomial, and RBF kernel. We chose the best kernel per class using cross-validation. The results for the VOC2009 are shown in Tab. 4. The mean average precision of our combination of the base classifiers is 53.09%, which compared to our base line means a performance increase of 2.95%.

Classification Experiments using Meta-class Information. We conducted initial experiments using automatically obtained meta-class for VOC2009 as described in 3.3 and adding these meta-class information as additional input in the Equation (2) of Section 3.2. The combination with additional meta-class information has a performance of 51.10%. The one-vs-all classifiers has a performance of 50.14%, therefore our method increases the performance by 0.96%. However, the simple combination (meta learners without meta-class information) method as described in Section 3.1 has a performance of 53.09%, which is our best result for the VOC2009 dataset. Our first experiments show that using additional classifiers from meta-classes give a performance loss of 1.99%. Additional experiments have to be done to investigate this outcome and also the effect of different tree structures on the performance. The reason for the decreased performance might be overfitting.

3.5 Conclusions and Open Questions

Our method of combining classifiers for multilabel classification is appealingly simple and works quite well. Our two layer approach can learn relevant interdependencies between classes. Therefore we got mean performance improvements of 9.53% (VOC2006) and 2.05% (VOC2009). We also showed that our automated construction of taxonomy trees based only on low-level features gives interpretable meta-classes. The main open question is whether the performance can be improved using meta-class information, but our results so far are negative.

class	one-vs-all	combination	impr.
aeroplane	77.0	79.5	2.5
bicycle	48.5	52.1	3.6
bird	53.3	57.2	3.9
boat	57.0	59.9	2.9
bottle	28.9	29.3	0.4
bus	63.9	63.5	-0.4
car	51.9	55.1	3.2
cat	52.1	53.9	1.8
chair	48.5	51.1	3.0
cow	30.8	31.3	0.5
diningtable	31.1	42.9	11.8
dog	43.6	44.1	0.5
horse	54.7	54.8	0.1
motorbike	55.5	58.4	2.9
person	77.2	81.1	3.9
pottedplant	19.8	30.0	10.2
sheep	36.0	40.2	4.2
sofa	46.3	44.2	-2.1
train	71.7	74.9	3.2
tv/monitor	54.9	58.2	3.3
avg	50.14	53.09	2.95

Table 4: Comparison of the classifier using a one-vs-all architecture, and our approach on the VOC 2009 dataset. Results are in percentage using the average precision measure.

4 Hierarchical Image-Region Labeling via Structured Learning

When classifying and segmenting images, some classes will be possible to identify based on global properties of the image, whereas others will depend on highly local information; many approaches deal with this problem by extracting features at multiple scales. However, segmentation based on local information can be highly noisy unless smoothness constraints are enforced. These two facts present a problem from a learning perspective: while it is possible to learn a first-order classifier which incorporates information from multiple scales, learning a classifier which enforces smoothness constraints is an example of *structured learning*, which appears to have made very little progress in this area due to the NP-hardness of many smoothness-enforcing algorithms.

In this section, therefore we will present a tree-structured graphical model that can be used to enforce smoothness constraints, while incorporating image features extracted at multiple scales. The tractability of this model will render it easily amenable to structured learning, which we believe is novel in this kind of segmentation scenario. We will define a loss based on approximate segmentations of an image (such as bounding boxes), allowing us to exploit the large amount of information in datasets such as VOC2007 and VOC2008 [9, 10]. We will use visual features from [6], which appear to exhibit state-of-the-art performance in classification problems, and show that their patch-level classification performance can be improved by structured learning.

The main contributions of this section are as follows: firstly, in contrast to many patch-level segmentation schemes, inference in our model is efficient and exact, allowing us to

circumvent the problems encountered when performing structured learning with approximate algorithms. Secondly, instead of parametrizing our model using image features directly, our model is parametrized using the outputs of a series of first-order classifiers, and can therefore easily extend and combine existing first-order classification approaches.

4.1 Related literature

The idea of partitioning an image into regions at multiple scales is certainly not novel [17, 25]. However, these papers are solely concerned with global categorization, whereas we also consider the problem of local region labeling. Furthermore, we shall not use the features of the image regions themselves, but rather our features are probability scores generated by existing first-order models, such as those from [6].

The approach of using tree-structured graphical models to incorporate global data into local segmentation problems (or simply to circumvent the problems associated with grid-structured models) is also not new; see for example [18]. However, to our knowledge, our approach is the first to apply structured-learning in this scenario, in order to improve the classification results of first-order classifiers. Similarly, others use information at an image-level to guide segmentation at the patch-level [37]. Other papers using similar approaches (though for different applications) include [33, 13, 39].

Many papers use lattice-structured Markov Random Fields (MRFs) to deal with the problem of patch-level segmentation. The unary and pairwise terms in such models may be similar to ours, i.e., the pairwise energy typically denotes a smoothness constraint. Energy minimization in such a model is NP-hard in the general case, so approximate forms of inference must be used [46]. There are many examples in this framework, though some important papers include [3] (α-*expansion,* $\alpha\beta$-*swap*), [36] (*normalized cuts*), and [26] (*log-cut*). [40] presents a comparative study of these ideas. We avoid such models as the need to perform approximate inference is of major concern from a learning perspective.[1]

Other authors also use grid structured models, but restrict their potential functions such that the energy minimization problem can be solved exactly. Examples include boolean-submodular functions [22, 2], *lattice*-submodular functions [21], and convex functions [19]. However, the energy functions we wish to use certainly do not fall into any of these categories.

4.2 Overview of the proposed graphical model $\mathcal{M}$

The nodes ($\mathcal{X}$) in our graphical model ($\mathcal{M}$) are similar to the regions used in [17, 25] (though in those papers, they do not form a graph); these nodes are depicted in Figure 4, and will be indexed by $x_{l,(i,j)}$ where l is the node's level in the graphical model, and (i,j) is its grid position. In fact, a similar graphical model has been suggested for binary segmentation in [27].

In words, a node on level k is connected to a node on level $k+1$, if and only if the regions corresponding to these nodes overlap. More formally,

$$x_{k,(i,j)} \text{ is connected to } \left\{x_{k+1,(2i,2j)}, x_{k+1,(2i,2j+1)}, x_{k+1,(2i+1,2j)}, x_{k+1,(2i+1,2j+1)}\right\} \quad (3)$$

(equivalently, $x_{k,(i,j)}$ is connected to $x_{k-1,(i/2,j/2)}$). Note that the graphical model is *undirected*. It is worth noting that there are no connections *between* nodes at a given level, and as such we are not enforcing neighborhood constraints *per se*. Neighborhood constraints will only be enforced indirectly, due to the fact that neighbors are connected via their parents.[2] Such a formulation is preferable because it results in a *tractable* graphical model (it forms

[1] Some papers make an exception to this rule, such as [5] and [35]. Recently, some theoretical results have been obtained regarding the performance of structured learning in such cases [14].

[2] In other words, we assume that two neighbors are *conditionally independent*, given their parents.

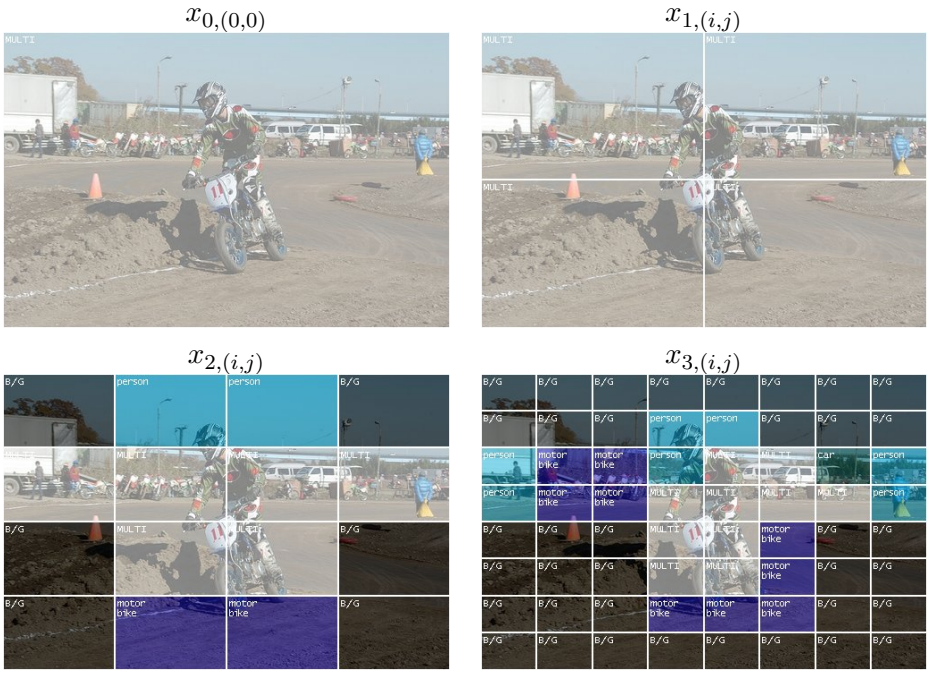


Figure 4: Nodes on the first four levels of our model. Nodes are indexed by $x_{l,(i,j)}$ where l is the node's level in the graphical model, and (i, j) is its grid position.

a quadtree), whereas enforcing neighborhood constraints directly typically requires that we resort to approximate forms of belief propagation.

The number of levels in this graph will depend on the size of the images in question, as well as how densely image patches are sampled. In practice, our graph is built to the maximal depth such that every region contains at least one patch, meaning that the 'regions' on the bottom level are essentially 'patches'. Details are given in Section 4.5.

4.2.1 Modeling hierarchical constraints in $\mathcal{M}$

When used in a classification scenario, this model will ensure that nodes on higher levels of the hierarchy (i.e., nodes with smaller values of l) will always be assigned to at least as generic a class as nodes on lower levels; this is precisely analogous to the notion of *inheritance* in object-oriented programming. The simplest inheritance diagram (denoted $\mathcal{H}$) that we may use is depicted in Figure 5.

Note that in $\mathcal{H}$, the class 'background' is actually a child of all specific classes; this is done because our training data may only approximately segment the image (i.e., using a bounding box). Thus an object that is classified as 'cat' on a higher level may be separated into 'cat' and 'background' on the level below. In principle, we could add additional classes to the hierarchy, representing 'clusters' of specific classes – for instance, tables and chairs may only appear in indoor scenes, whereas sheep and horses may only appear outdoors. The problem of learning such a visual hierarchy has been addressed in [38] (among others), and incorporating such hierarchies is certainly an avenue for future work.

These requirements will be enforced as *hard* constraints in $\mathcal{M}$ (i.e., assignments which disobey these constraints will have cost ∞). We will use the notation $a \prec b$ to indicate that a class a is *less specific* than b.

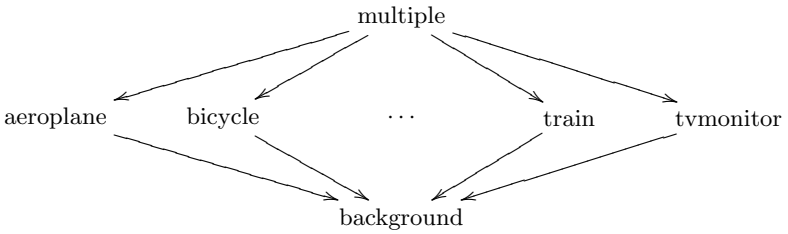


Figure 5: The simplest possible hierarchy; the most general possible assignment simply states that an object may belong to any of multiple classes; the least general states that an object does not belong to any class (↓ denotes 'more general than'). Class labels are taken from [9, 10], though any set of labels could be used.

4.3 Probability maximization in $\mathcal{M}$

4.3.1 Unary potentials.

In the most general case, the unary potentials (i.e., first order probabilities) in $\mathcal{M}$ are defined as follows (note that we have suppressed the subscript of the region x for brevity):

$$E(x;y) = \left\langle \Phi^1(x,y), \theta^{\text{nodes}} \right\rangle, \tag{4}$$

where $y \in \mathcal{H}$ is the class label to which the region x is to be assigned, and θ^{nodes} is a feature vector parametrizing $\Phi^1(x,y)$ – the *joint feature map* of the node x and its assignment y.

In our specific case, we wish to ensure that the class label given to x was given a high probability according to some first-order classifier (such as that defined in [6]). Specifically, suppose that such a classifier returns a vector of probabilities P_x; then our joint feature map may be

$$\Phi^1(x,y) = \underbrace{(0,\ldots,P_{x,y},\ldots,0)}_{\text{0 everywhere except the } y^{\text{th}} \text{ entry}} . \tag{5}$$

In words, if x is assigned the class y, then the probability given by the first-order model should be high (weighted by θ_y^{nodes}).[3]

There are two straightforward generalizations of this model which may be desirable: firstly, we may wish to learn a separate parametrization for each level of the hierarchy. In this case, we would have a copy of $\Phi^1(x,y)$ for each level of the hierarchy, and use an indicator function to 'select' the current level. The second generalization would be to parametrize *multiple* first-order classifiers, which return probabilities $P_x^1 \cdots P_x^C$ (for instance, features based on histograms of orientations; features based on RGB statistics, etc.). In this case, our joint feature map will simply be a concatenation of the individual feature maps defined by each classifier (i.e., it will be nonzero in exactly C locations). We will use both of these generalizations in our experiments (see Section **??**).

4.3.2 Pairwise potentials.

Similarly, we define pairwise potentials for nodes x_k and x_{k+1} (suppressing the remainder of the index):

$$E(x_k, x_{k+1}; y_k, y_{k+1}) = \left\langle \Phi^2(x_k, x_{k+1}; y_k, y_{k+1}), \theta^{\text{edges}} \right\rangle. \tag{6}$$

This time the joint feature map Φ^2 should express two properties: firstly, the constraints of our hierarchy should be strictly enforced; secondly, nodes assigned to the same class on

[3] To simplify, $E(x;y) = P_{x,y}\theta_y$. The formulation in (eq. 5) is used simply to express this as a linear function of θ^{nodes}. Also, we use $\log(P_{x,y})$ in practice, since we are maximizing a sum rather than a product.

different levels should have similar probabilities (again using the probabilities P_{x_k} and $P_{x_{k+1}}$ returned by our first-order classifier).

To achieve these goals, we define the indicator function H as

$$H(y_k, y_{k+1}) = \begin{cases} \infty & \text{if } y_k \succ y_{k+1}, \\ 0 & \text{if } y_k \prec y_{k+1}, \\ 1 & \text{otherwise } (y_k = y_{k+1}). \end{cases} \tag{7}$$

Note that this indicator function enforces precisely the hierarchical constraints that we desire. It also specifies that there is no cost associated to assigning a child node to a more specific class – thus we are only parametrizing the cost when both class labels are the same. Our joint feature map now takes the form

$$\Phi^2(x_k, x_{k+1}; y_k, y_{k+1}) = -H(y_k, y_{k+1})\left|P_{x_k} - P_{x_{k+1}}\right|^2, \tag{8}$$

where $|p|$ is the *elementwise* absolute value of p. Again we may make the same extensions to this model as outlined in Section 4.3.1.

4.3.3 The potential function

The complete maximization function is now defined as

$$g_\theta(\mathcal{X}) = \operatorname*{argmax}_{\mathcal{Y}} \sum_{x \in \mathcal{M}} \left\langle (0, \ldots, P_{x,y(x)}, \ldots, 0), \theta^{\text{nodes}} \right\rangle + \sum_{x_k, x_{k+1} \in \mathcal{M}} \left\langle H(y_k(x_k), y_{k+1}(x_{k+1}))\left|P_{x_k} - P_{x_{k+1}}\right|^2, \theta^{\text{edges}} \right\rangle, \tag{9}$$

where θ is simply the concatenation of our two feature vectors $(\theta^{\text{nodes}}; \theta^{\text{edges}})$, and $y(x)$ is the assignment given to x under $\mathcal{Y}$ (the full set of labels). As the nodes in $\mathcal{M}$ form a tree, this energy can be maximized using the *Junction Tree Algorithm* (see, for example [1]). The running time of this procedure is in $O(|\mathcal{M}||\mathcal{H}|^2)$, where $|\mathcal{M}|$ is the number of nodes and $|\mathcal{H}|$ is the number of classes.

4.3.4 Loss function

Our loss function specifies 'how bad' a given assignment $\mathcal{X}$ is compared to the correct assignment $\mathcal{Y}$. We desire that our loss function should decompose as a sum over nodes and edges in $\mathcal{M}$ (for reasons shown in Section 4.3.5).

Firstly, we must specify how our training labels $\mathcal{Y}$ are produced using existing datasets. One option is simply to assign the class 'multiple' to all regions with which multiple bounding boxes intersect, to assign 'background' to all regions with which *no* bounding boxes intersect, and to assign a specific class label to all others. Specifically, we will define a loss function of the form

$$\Delta(\mathcal{X}, \mathcal{Y}) = \sum_{i=1}^{|\mathcal{X}|} \delta(x_i, y_i). \tag{10}$$

One such loss is the *Hamming loss*, which simply takes the value 0 when the region is correctly assigned, and $1/|\mathcal{M}|$ otherwise, where $|\mathcal{M}|$ is the number of regions (or more formally $\delta(x_i, y_i) = \frac{1}{|\mathcal{M}|}(1 - I_{\{x_i\}}(y_i))$).[4] In practice, we scale the loss so that each level of our graphical model makes an equal contribution (i.e., a mistake on level k makes four times the contribution as a mistake on level $k+1$).

[4] When multiple classes are observed in a single region (i.e., when the correct label is 'multiple'), no penalty is incurred if one of these specific classes is chosen.

4.3.5 Structured learning in $\mathcal{M}$

Here we present a brief overview of the Structured Learning framework described in [42], and demonstrate that the current application can be expressed in this framework.

4.3.6 The risk functional

The risk functional that we would like to optimize (over the parameter vector θ) takes the form

$$\operatorname*{argmin}_{\theta} \Big[\underbrace{\frac{1}{N} \sum_{n=1}^{N} \Delta(g_\theta(\mathcal{X}^n), \mathcal{Y}^n)}_{\text{empirical risk}} + \underbrace{\lambda \|\theta\|^2}_{\text{regularization term}} \Big]. \tag{11}$$

where g is our (θ-parametrized) function which classifies the training sample $\mathcal{X}^n$, $\mathcal{Y}^n$ is the correct class of $\mathcal{X}^n$, and $\Delta(y^*, y^n)$ is our *loss function*, encoding the cost of making the classification y^* when the correct class is y^n.[5] These terms have all been defined for the present application in Section 4.3.

In the case of an SVM, the loss function Δ would simply be a 0/1 loss (i.e. 0 if the classification is correct, 1 if not). In the present case, we use the loss described in Section 4.3.4. This highlights a very important difference between the present model and an SVM: note that for a binary classifier (in the separable case), the empirical risk can be decomposed into a set of constraints which are linear in θ, resulting in a convex optimization problem (since the regularization term is typically convex). However, in the case of structured classifiers, the loss is piecewise constant in θ, and is certainly not convex. Thus we cannot hope to solve (eq. 11) exactly in the structured case.

4.3.7 The soft-margin formulation

The soft-margin formulation of structured learning is again very similar to that of SVMs:

$$\text{minimize}_{\theta,\xi} \ \frac{1}{N} \sum_{n=1}^{N} \xi_n + \lambda \Omega(\theta) \tag{12a}$$

$$\text{subject to } \langle \Phi(\mathcal{X}^n, \mathcal{Y}^n) - \Phi(\mathcal{X}^n, y), w \rangle \geq \Delta(y, \mathcal{Y}^n) - \xi_n \quad \text{for all } n \text{ and } y \in \mathcal{S}, \tag{12b}$$

where $\mathcal{S}$ is the space of all possible labellings for our training samples. Note that this is equivalent to the soft-margin formulation of SVMs, but for two differences: firstly there is an additional term $\Phi(\mathcal{X}^n, \mathcal{Y}^n)$ (the 'potential' of choosing the correct assignment, as defined in Section 4.3.3).[6] Secondly, the number of constraints in the optimization problem is greatly increased: while there were only N constraints in the SVM formulation, there are now $N|\mathcal{S}|$. This is clearly a problem, since the space in which we are working may be exponentially large. Also, given the piecewise constancy of Δ in the structured case, we are no longer minimizing (eq. 11) exactly, but instead we are minimizing a *convex upper bound* on the risk.

4.3.8 Column generation

Since we are naturally unable to include all of the constraints in (eq. 12), we seek a procedure which helps us choose the 'best' constraints; this procedure is known as *column generation*.

[5] N is the number of training samples; $\Omega(\theta)$ is a regularization term (such as the l^2 norm of θ); the hyperparameter λ controls the importance of this term, and is in practice selected using our validation set.

[6] Actually, this term *does* appear in the SVM formulation, but it is factored into the terms denoted b and y_i in [4].

We wish to find the most violated constraints in (eq. 12), and add these to the optimization problem. In order to do so, we need to solve (for each n)

$$\hat{y}_n = \underset{y}{\operatorname{argmax}} \left[\langle \Phi(\mathcal{X}^n, y), w \rangle + \Delta(y, \mathcal{Y}^n) \right], \tag{13}$$

as this is the term for which the constraint in (eq. 12b) is tightest (i.e. the constraint that maximizes ξ_n). Typically, in order for this problem to be tractable, we require that Δ is decomposable so that this function can be optimized using the same procedure used to optimize (eq. 9).[7]

Assuming these requirements are satisfied, the problem will now be solvable by standard optimization procedures, which we will not cover here.

4.4 Structured Learning in the Present Framework

The framework described in [42] and above requires that we are able to solve (in addition to (eq. 9) itself)

$$g_\theta(\mathcal{X}) = \underset{\mathcal{Y}}{\operatorname{argmax}} \sum_{x \in \mathcal{M}} \left\langle (0, \ldots, P_{x,y(x)}, \ldots, 0), \theta^{\text{nodes}} \right\rangle + \sum_{x_k, x_{k+1} \in \mathcal{M}} \left\langle H(y_k(x_k), y_{k+1}(x_{k+1})) \left| P_{x_k} - P_{x_{k+1}} \right|^2, \theta^{\text{edges}} \right\rangle + \Delta(\mathcal{X}, \mathcal{Y}), \tag{14}$$

As mentioned, this is possible as long as $\Delta(\mathcal{X}, \mathcal{Y})$ decomposes as a sum over the nodes and edges in our model (which is certainly true of the Hamming loss).

4.5 Experiments

We used images from the VOC2007 and VOC2008 datasets to evaluate our model. Specifically, we used VOC2008 for training and validation, and VOC2007 for testing (as testing data for VOC2008 is not available). This presents a learning scenario in which there is a realistic difference between the training and test sets. The training, validation, and test sets contain 2113, 2227, and 4952 images respectively. The validation set is used to choose the optimal value of the regularization constant λ.

We extracted SIFT-like features for our model on uniform grids at 5 different scales [28]. We used the methodology of [6] based on Fisher vectors to extract a signature for each region. A patch is considered to belong to a region if its centre belongs to that region, and its overlap with the region is at least 25%. We used three different first-order classifiers, based on sparse logistic regression: one which has been trained to classify the entire collection of features in an image (the 'image-level' classifier), one which has been trained on bounding-boxes (the 'mid-level' classifier), and one which has been trained on individual patches (the 'patch-level' classifier). The baseline to which we compare our method is one which simply selects the highest score using these individual classifiers (i.e., no consistency information is to be used). This baseline is similar to what is reported in [6], though it is important to stress that their method was not optimized to minimize the same loss that is presented here. We also report the performance using the image prior defined in [6], which rejects labellings at the patch level which are inconsistent with the probability scores at the image level.

Classification scores for the classes 'background' and 'multiple' were extracted automatically from the first-order scores: the probability of belonging to the background is 1 minus the highest probability of belonging to any other class; the probability of belonging to multiple

[7] To be completely precise, (eq. 13) must be solved at *training* time, whereas (eq. 9) must be solved only at *testing* time.

	Training	Validation	Testing
Baseline (see [6])	0.272 (0.004)	0.273 (0.004)	0.233 (0.003)
Baseline with image prior (see [6])	0.275 (0.005)	0.276 (0.004)	0.238 (0.003)
Non-learning	0.235 (0.006)	0.224 (0.005)	0.233 (0.004)
Learning	**0.460** (**0.006**)	**0.456** (**0.006**)	**0.374** (**0.004**)

Table 5: Performance of our method during training, validation, and testing (for the optimal value of λ), compared to non-learning methods. The value reported is simply the proportion of correctly labeled regions, with each level contributing equally (i.e., one minus the loss). Values in parentheses indicate the standard error (across all images).

	Level 0	Level 1	Level 2	Level 3
Baseline (see [6])	0.342	0.214	0.232	0.163
Baseline with image prior (see [6])	0.342	0.217	0.242	0.169
Non-learning	**0.426**	0.272	0.137	0.112
Learning	0.413	**0.307**	**0.349**	**0.444**

Table 6: Performance on each level of the graph (on the test set).

classes is the twice the product of the two highest probabilities, capped at 1 (so that if two classes have probabilities greater than 0.5, the product will be greater than 0.5 also).

Structured learning was performed using the 'Bundle Methods for Risk Minimization' code of [41]. This solver requires only that we specify our feature representation $\Phi(\mathcal{X}, \mathcal{Y})$ (eq. 5, 8), our loss $\Delta(\mathcal{X}, \mathcal{Y})$ (eq. 10), and a column-generation procedure (eq. 14).

A performance comparison between the learning and non-learning versions of our approach, as well as the baseline is shown[8] in Table 5. Figure 6 shows[9] an example match from our test set. Table 6 shows the contribution to the loss made by each level of the graphical model and Figure 7 shows the weight vector learned by our method. Note that our model exhibits a substantial improvement over the baseline, and non-learning approaches.

Further qualitative results are shown in Figures 8 and 9. In Figure 9 we have some example cases in which our model performs poorly. We observed that much of our method's improvement comes as a result of decreasing the number of false-negatives in the 'person' class, but at the cost of increasing the number of false-positives; ultimately, this leads to a reduction in the total loss, since the 'person' class is by far the most common in the datasets being used [9, 10]. Consequently, objects which frequently co-occur with (or whose bounding boxes overlap with) people are frequently misclassified as people (such as bicycles, bottles, dining tables). It is important to stress that we could choose a different loss function to avoid this behaviour: for instance, one which assigned a higher penalty to false-negatives than false-positives, or one which assigned a smaller weight to the 'person' class. However, in a dataset where people are by far the most common class, we believe that the observed behaviour is sensible.

Comparison with other methods is certainly difficult, as our loss is neither equivalent to a classification nor a segmentation error. Note however that in Table 6, we achieve an improvement at *both* the segmentation and classification levels. Also, due to the class 'multiple', our loss is *not* equivalent to the criteria normally used to measure performance on the VOC2007 and VOC2008 datasets.

It is difficult even to provide a fair comparison to the method of [6], as that model was

[8]The non-learning version of the approach just sets $\theta = (1, 1, \ldots, 1, 1)$, though any constant value will do.

[9]Figures cited in this section can be found in the **Appendix.**

	Level 0	Level 1	Level 2	Level 3
Baseline with image prior ($k = 2$)	**0.342**	**0.217**	0.242	0.169
Baseline with image prior ($k = 3/2$)	0.330	0.214	0.245	0.184
Baseline with image prior ($k = 4/3$)	0.324	0.211	**0.246**	0.185
Baseline with image prior ($k = 1$)	0.310	0.195	0.241	**0.186**

Table 7: Comparison of the constant factor k by which the two highest probabilities are multiplied to produce the probability for the class 'multiple'.

	Training	Validation	Testing
Baseline (see [6])	0.293 (0.005)	0.295 (0.005)	0.248 (0.003)
Baseline with image prior (see [6])	0.299 (0.005)	0.301 (0.005)	0.254 (0.003)
Non-learning	0.238 (0.005)	0.228 (0.005)	0.231 (0.004)
Learning	**0.473** (**0.006**)	**0.469** (**0.006**)	**0.382** (**0.004**)

Table 8: Performance of our method during training, validation, and testing (for the optimal value of λ), using 3×3 branching. Note that the performance shown is not directly comparable to that of 2×2 branching, as different regions are used to compute the loss.

not optimized to minimize the same loss, nor was it designed to deal with the 'multiple' class. For this reason, the probability chosen for the class 'multiple' is very important reporting the performance of their method. As mentioned in the previous section, this probability is defined to be $k \times p_1 \times p_2$ was chosen, where p_1 and p_2 were the two highest probabilities amongst all individual classes. The constant $k = 2$ was chosen in the previous section so that whenever p_1 and p_2 are greater than 0.5, the probability for 'multiple' will be above 0.5 also. In Table 7 and Figure 10 we show the effect of choosing different values for k in this expression. Although the results are visually quite different, the difference in their performance is small.

4.6 Using 3×3 split in our model

One simple modification to our model is to split each region into a different number of subregions at each level. In the previous section, each region was split into 2×2 subregions; in Figure 11 we show that a 3×3 split is also possible, and give some results for comparison.

Tables 8 and 9 show the performance of this method compared to the baseline, and Figure 12 shows the weight vector that was learned. It is important to note that the values of the loss shown here are not directly comparable to the values shown when using a 2×2 split, as the image regions on which the loss is computed are different. Visually, the results of the 3×3 split appear to be inferior to those of the 2×2 split shown in the previous section.

4.7 Conclusion

We have presented a graphical model which efficiently performs patch-level, region-level, and image-level labeling simultaneously. This model is useful in that it allows us to encode smoothness constraints for patch-level classification, while still incorporating the important information at higher levels. We have shown how to apply structured learning in this model, which has traditionally been a problem for models incorporating smoothness constraints. We have shown that our model improves in performance over existing models which use only first-order information. Since our model is parametrized using the probability scores from first-order approaches, it should be seen as complimentary to existing first-order techniques.

	Level 0	Level 1	Level 2
Baseline (see [6])	0.342	0.248	0.186
Baseline with image prior (see [6])	0.342	0.255	0.196
Non-learning	**0.426**	0.173	0.123
Learning	0.410	**0.314**	**0.452**

Table 9: Performance on each level of the graph using 3×3 branching (on the test set). Note that the graph is generally less deep when using 3×3 branching, and was never more than three levels deep in our training set (hence only three levels are shown).

The experiments wer done on a relatively small dataset (PASCAL VOC2008/2007), which are relatively small for the envisaged application scenarios. However, these are recent benchmark datasets which have been explored by state-of-the-art methods (see [6] and the segmentation results present in [10]). These datasets are in fact unique in terms of the number of labelled bounding boxes provided and the consistency of the labels. Since the proposed method is inductive, it is easily scalable to large datasets (in terms of number of images). In terms of number of categories, the scalability depends on the scalability of the first order classifiers. We use linear sparse logistic regression [23] as first order classifiers, meaning that the training set size has no impact on the computational cost for inference. The hierarchical constraints proposed here should reduce the impact of the number of classes in the accuracy of region classification. To learn the structure weights, an increase in the number of classes implies in an $O(c^2)$ increase in training time. For inference, the computational cost is linear (O(c)) in the number of classes.

References

[1] S.M. Aji and R.J. McEliece. The generalized distributive law. *IEEE Trans. on Information Theory*, 46(2):325–343, 2000.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. on PAMI*, 26(9):1124–1137, 2004.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on PAMI*, 23(11):1222–1239, 2001.

[4] C. Cortes and V. Vapnik. Support vector networks. In *Machine Learning*, pages 273–297, 1995.

[5] T. Cour, N. Gogin, and J. Shi. Learning spectral graph segmentation. In *AISTATS*, 2005.

[6] Gabriela Csurka and Florent Perronnin. A simple high performance approach to semantic segmentation. In *Proc 19th British Machine Vision Conf, Leeds*, 2008.

[7] Teofilo E. de Campos, Florent Perronnin, and Gabriela Csurka. Images as sets of weighted features. In *Proc 20th British Machine Vision Conf, London*, September 2009.

[8] Sašo Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54(3):255–273, 2004.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[11] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html.

[12] Mark Everingham, Andrew Zisserman, Christopher K. I. Williams, and Luc J. Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[13] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[14] T. Finley and T. Joachims. Training structural SVMs when exact inference is intractable. In *ICML*, 2008.

[15] Michael Fussenegger, Andreas Opelt, Axel Pinz, and Peter Auer. Object recognition using segmentation for feature detection. In *International Conference on Pattern Recognition (ICPR) (3)*, pages 41–44, 2004.

[16] Luc J. Van Gool, Theo Moons, and Dorin Ungureanu. Affine/photometric invariants for planar intensity patterns. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision – Volume I*, pages 642–651. Springer, 1996.

[17] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005.

[18] X. He, R.S. Zemel, and M.Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. *CVPR*, 2004.

[19] H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Trans. on PAMI*, 25(10):1333–1336, 2003.

[20] Yan Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition (CVPR 2004)*, volume 2, pages 506–513. IEEE Computer Society, 2004.

[21] P. Kohli, A. Shekhovtsov, C. Rother, V. Kolmogorov, and P. Torr. On partial optimality in multi-label MRFs. In *ICML*, 2008.

[22] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *ECCV*, 2002.

[23] B. Krishnapuram, L Carin, M. Figueiredo, and A. Hastemink. Sparse multimodal logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968, June 2005.

[24] Jorma Laaksonen, Markus Koskela, Sami Laakso, and Erkki Oja. PicSOM - Content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21(13-14):1199–1207, December 2000.

[25] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[26] V. Lempitsky, C. Rother, and A. Blake. Logcut: Efficient graph cut optimization for markov random fields. In *ICCV*, 2007.

[27] J. Li, R.M. Gray, and R.A. Olshen. Multiresolution image classification by hierarchical modeling with two-dimensional hidden markov models. *IEEE Trans. on Information Theory*, 46(5):1826–1841, 2000.

[28] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004.

[29] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[30] Julian McAuley, Teofilo E. de Campos, Gabriela Csurka, and Florent Perronnin. Hierarchical image-region labeling via structured learning. In *Proc 20th British Machine Vision Conf, London*, September 2009.

[31] Julian McAuley, Teofilo E. de Campos, Gabriela Csurka, and Florent Perronnin. Hierarchical image-region labeling via structured learning. US Patent ???, July 2009. Xerox Corporation.

[32] Julian J. McAuley, Teófilo E. de Campos, and Tiberio Caetano. Unified graph matching in euclidean spaces. In *Proc of the Neural Information Processing Systems (NIPS)*, 2009.

[33] D. Ramanan and D.A. Forsyth. Finding and tracking people from the bottom up. *CVPR*, 2003.

[34] Ryan M. Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[35] D. Scharstein and C. Pal. Learning conditional random fields for stereo. *CVPR*, 2007.

[36] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 22(8):888–905, 2000.

[37] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *IJCV*, 81(1):2–23, 2009.

[38] J. Sivic, B.C. Russell, A. Zisserman, W.T. Freeman, and A.A. Efros. Unsupervised discovery of visual object class hierarchies. In *CVPR*, 2008.

[39] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Trans. on PAMI*, 28(9):1372–1384, 2006.

[40] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. on PAMI*, 30(6):1068–1080, 2008.

[41] C.H. Teo, Q. Le, A.J. Smola, and S.V.N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *KDD*, 2007.

[42] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Predicting Structured Data*, pages 823–830, 2004.

[43] Koen van de Sande, Theo Gevers, and Cees Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 99(1), 5555.

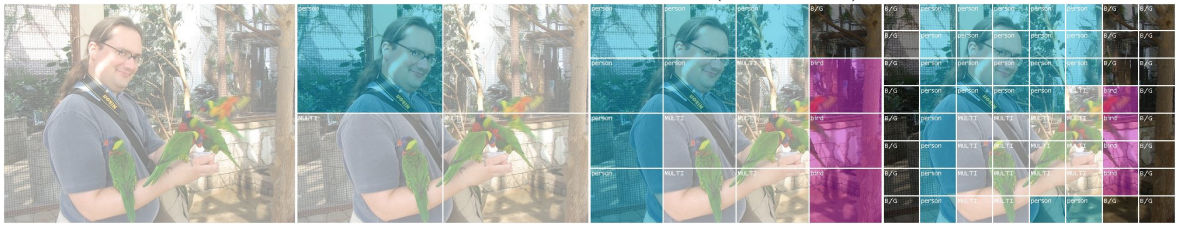
[44] Ville Viitaniemi and Jorma Laaksonen. Evaluation of pointer click relevance feedback in PicSOM (D1.2). Technical report, PinView European Comunity project FP7-216529, July 01 2008. Available at http://www.pinview.eu.

[45] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

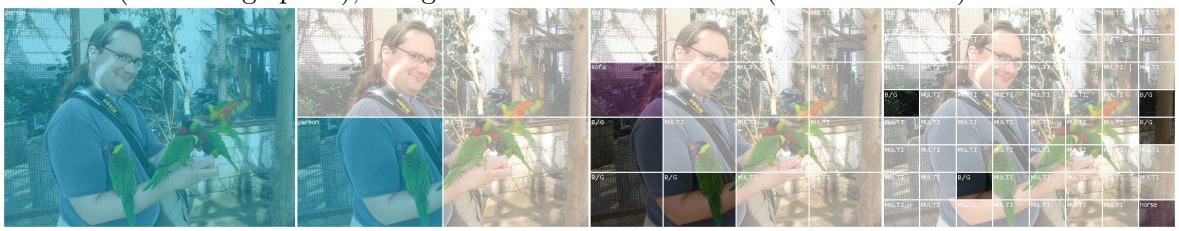
[46] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, 2000.

Appendix: Figures

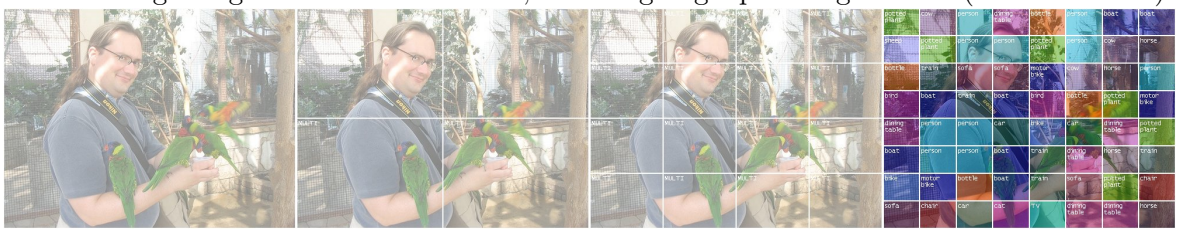
Correct labeling, using bounding-boxes from VOC2007 ($1 - \Delta = 1$):



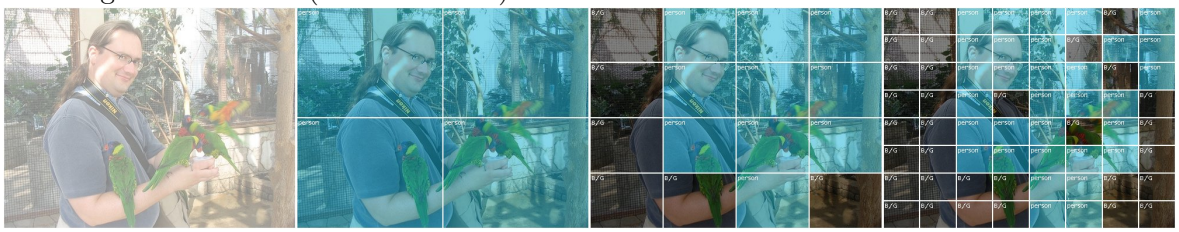
Baseline (with image prior), using no second-order features ($1 - \Delta = 0.566$):



Non-learning using second-order features, but assigning equal weight to all ($1 - \Delta = 0.551$):



Learning of all features ($1 - \Delta = 0.770$):



Colour-code for labels observed in these images:

Figure 6: An example match comparing our technique to non-learning methods. The top sequence contains the 'correct' labeling, as extracted from the VOC2007 dataset (Image 000127; the correct labeling incurs zero loss by definition). The second sequence uses only first order features (i.e., the most likely assignment is chosen for each region independently); the image-level classifier is used at the top level, the mid-level classifier is used at the second and third levels, and the patch-level classifier is used at the bottom level; the fact that many of regions at the bottom level are incorrectly labeled demonstrates the need for consistency constraints. The third sequence shows our method without learning (i.e., assigning equal weight to all features); the low quality of this match demonstrates that the weights are poorly tuned without learning. Finally, the fourth sequence shows the performance of our method, which appears to address both of these issues. A key for the labels used is also shown.

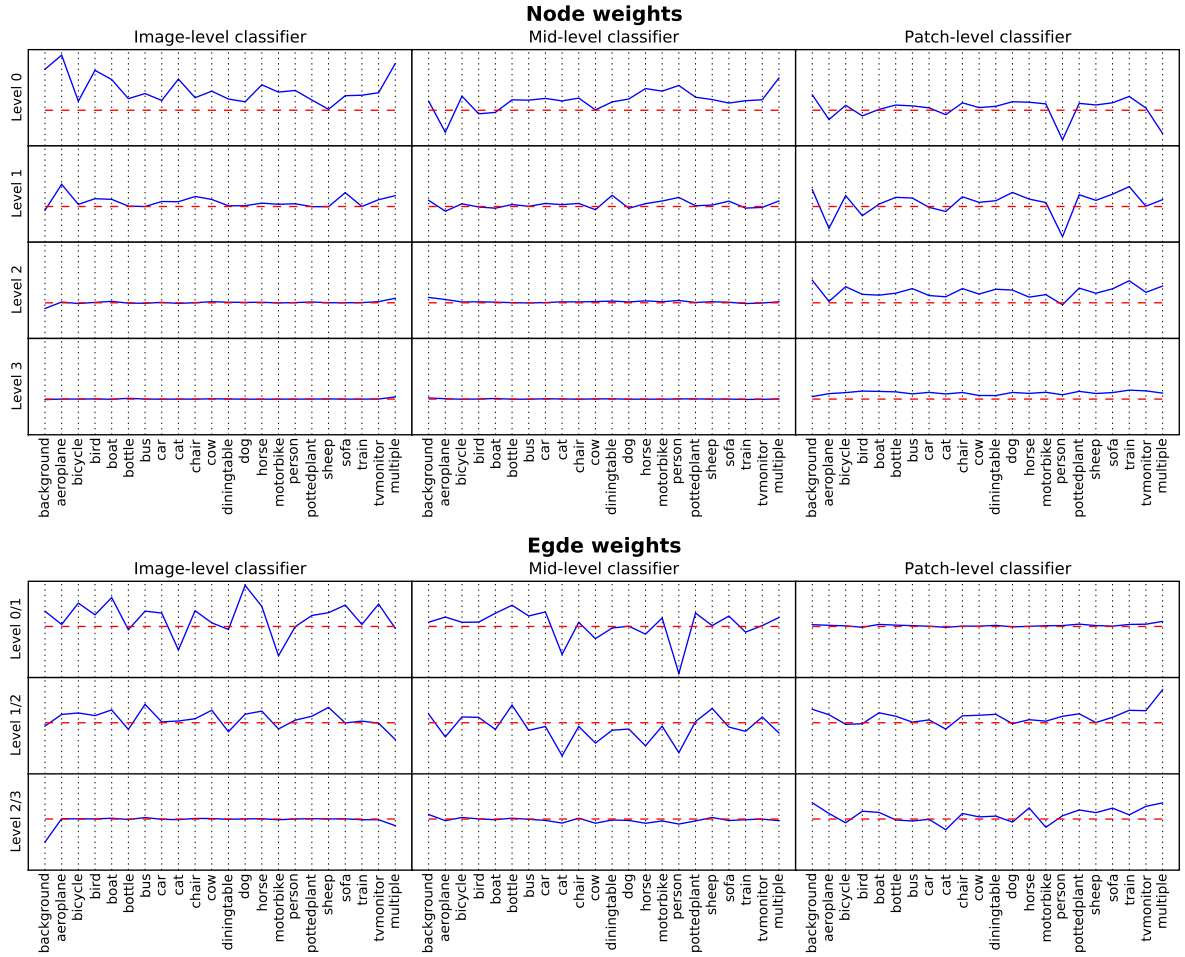


Figure 7: The complete weight-vector for our model. A separate vector of 22 ($= |\mathcal{H}|$) weights is learned for each image level, and for each type of classifier being used (the dashed line corresponds to zero). This vector has several interesting properties: note that the image-level classifier is given higher importance at the top levels, whereas the patch-level classifier is given higher importance at the bottom levels (which is consistent with our expectations). Also, note that there is a lot of variance between weights of different classes (especially 'multiple', 'background', and 'person'), indicating that certain classes are easier to identify at different scales. Finally, note that the edge weights have both large positive and negative scores: for instance, the high weight given to 'dog' for the image-level classifier at level 0/1 indicates that the features between these two levels should be very similar, whereas the *negative* weight given to 'cat' (at the same level) indicates that the features between the two levels should be very *different*.

Image 002475 ($1 - \Delta = 0.664$):

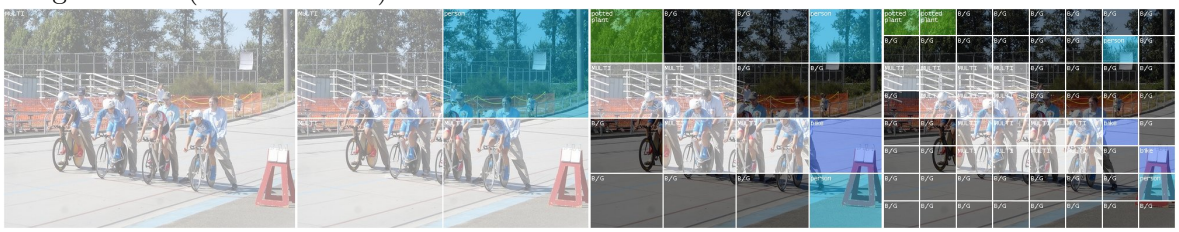


Image 002631 ($1 - \Delta = 0.676$):

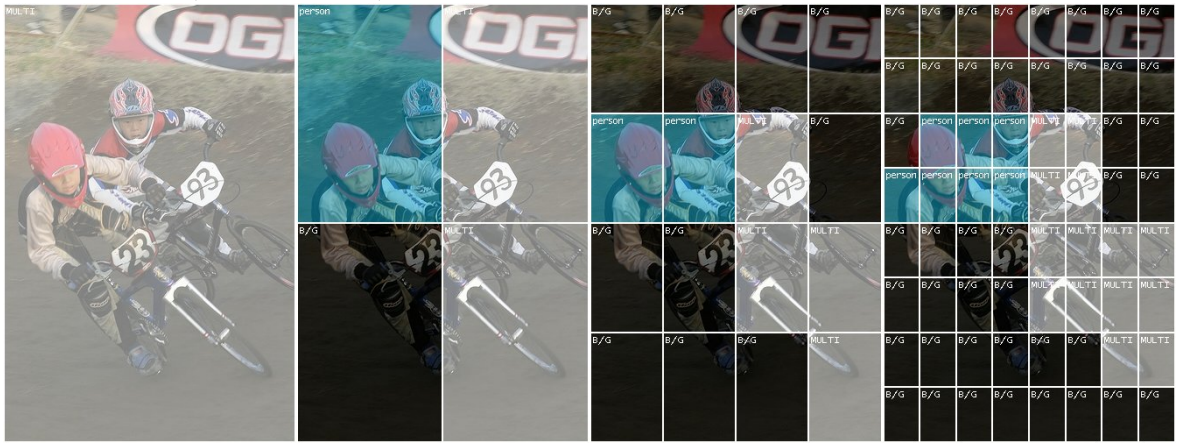


Image 003494 ($1 - \Delta = 0.680$):

Image 004225 ($1 - \Delta = 0.695$):

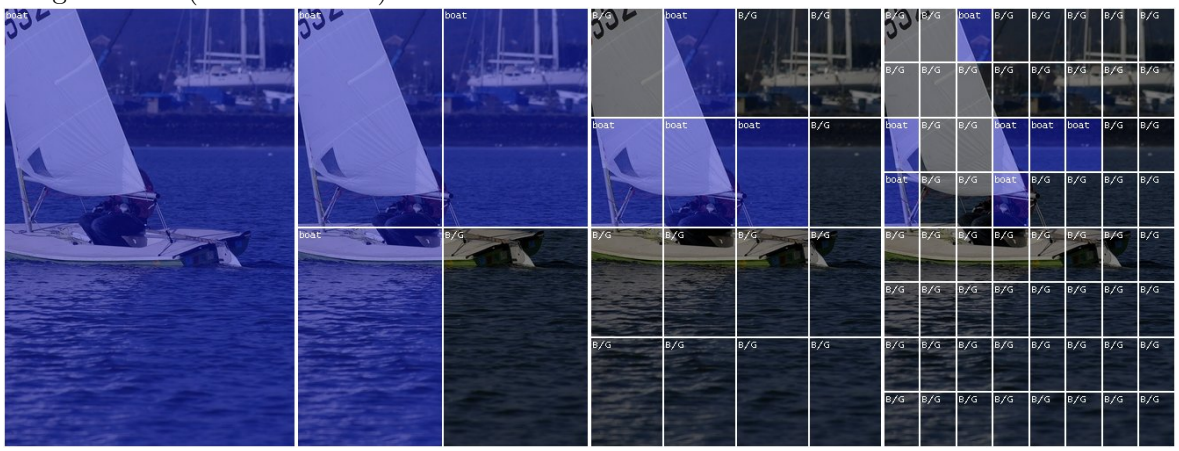


Figure 8: Some additional results using our method. Results are from the test set.

Image 002789: Objects are incorrectly identified as ‘background’ ($1 - \Delta = 0.637$):

Image 004410: The class ‘tvmonitor’ is mistaken for ‘person’ ($1 - \Delta = 0.617$):

Image 002856: Objects are misclassified as ‘person’ ($1 - \Delta = 0.676$):

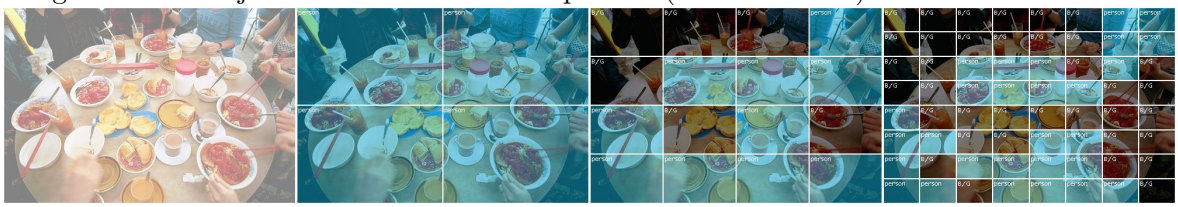


Image 003595 ($1 - \Delta = 0.812$):

Image 004765 ($1 - \Delta = 0.637$):

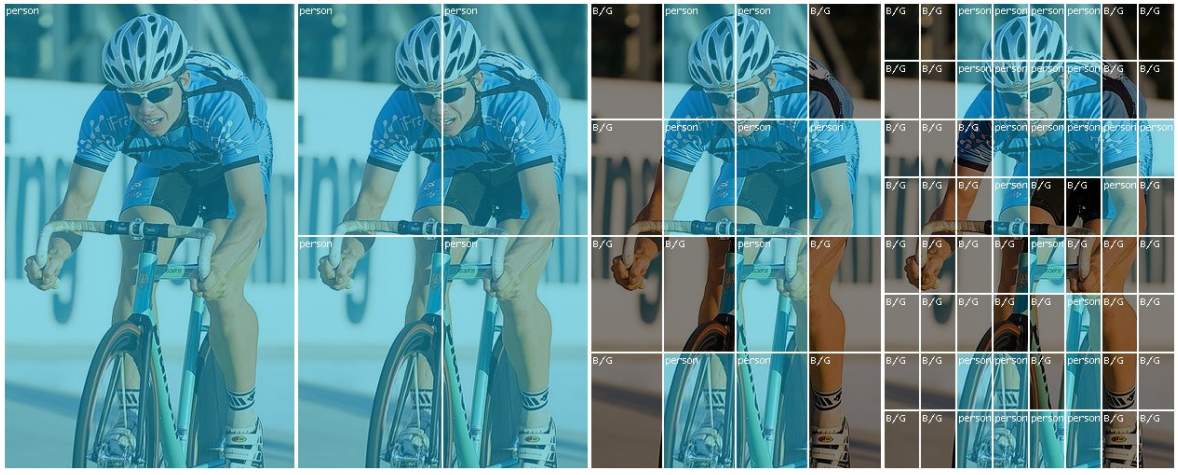
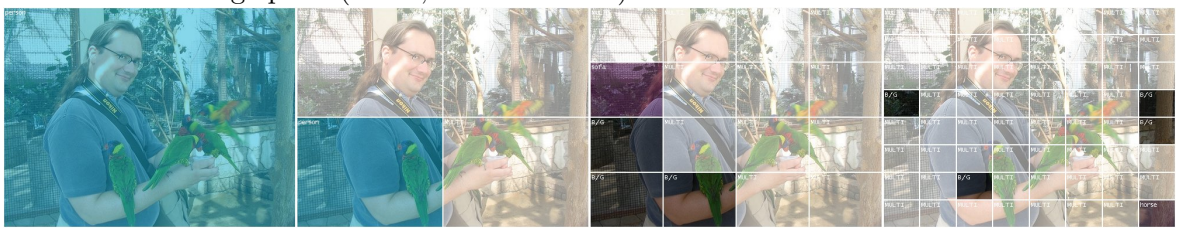
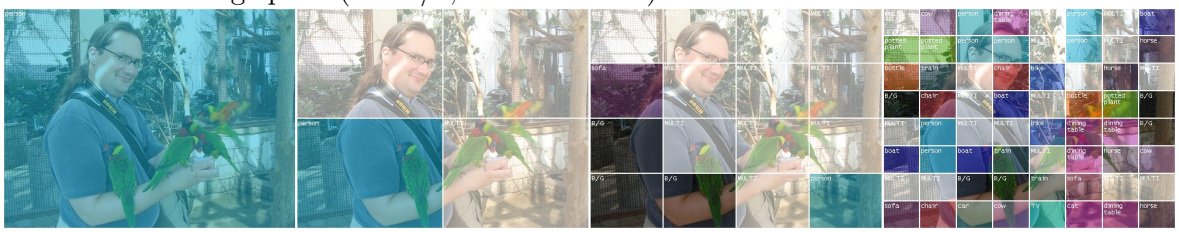


Figure 9: Some ‘failure cases’ of our method. Much of our method’s improvement comes from decreasing the number of false-negatives in the ‘person’ class; as a result, our method seems to increase the number of false-positives for this class.

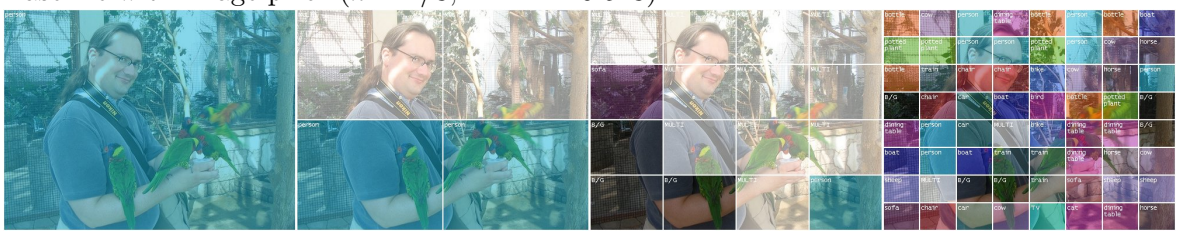
Baseline with image prior ($k = 2$, $1 - \Delta = 0.566$):



Baseline with image prior ($k = 3/2$, $1 - \Delta = 0.551$):



Baseline with image prior ($k = 4/3$, $1 - \Delta = 0.543$):



Baseline with image prior ($k = 1$, $1 - \Delta = 0.555$):

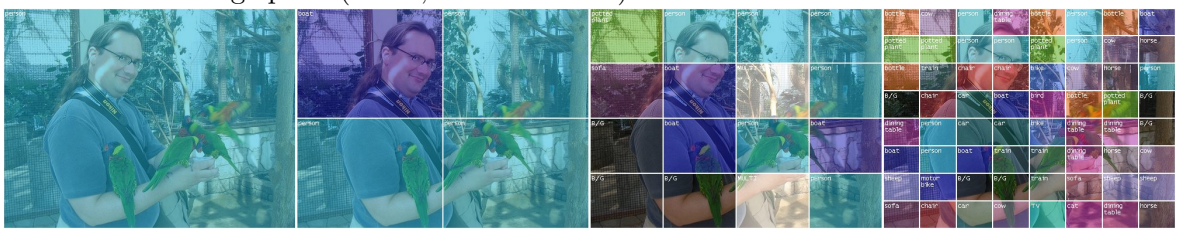


Figure 10: Comparison of the constant factor k. Note that it is still possible to observe the class multiple even when $k = 1$, due to the presence of the image prior.

Image 000127: Using 3×3 branching ($1 - \Delta = 0.794$):

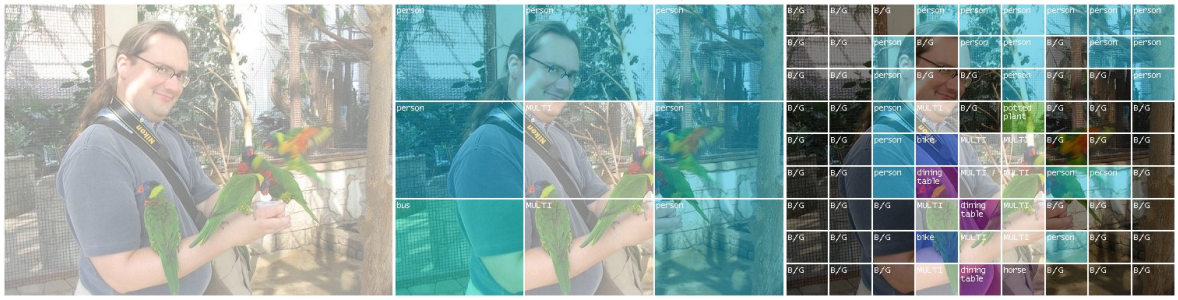


Image 000181 ($1 - \Delta = 0.852$):

Image 000643 ($1 - \Delta = 0.765$):

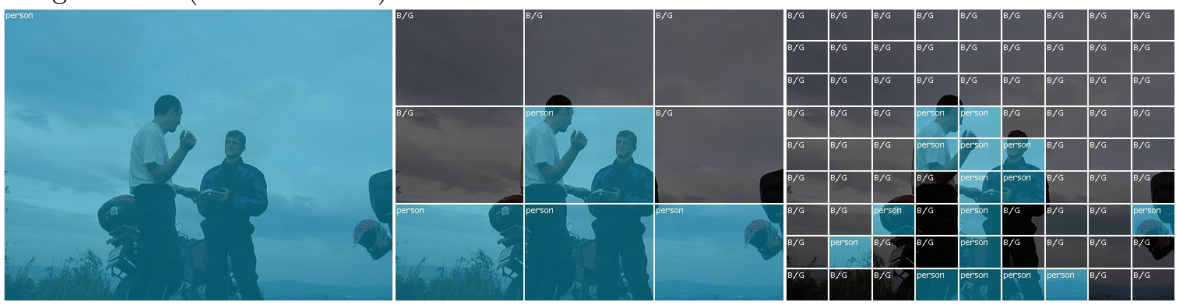


Image 001511 ($1 - \Delta = 0.724$):

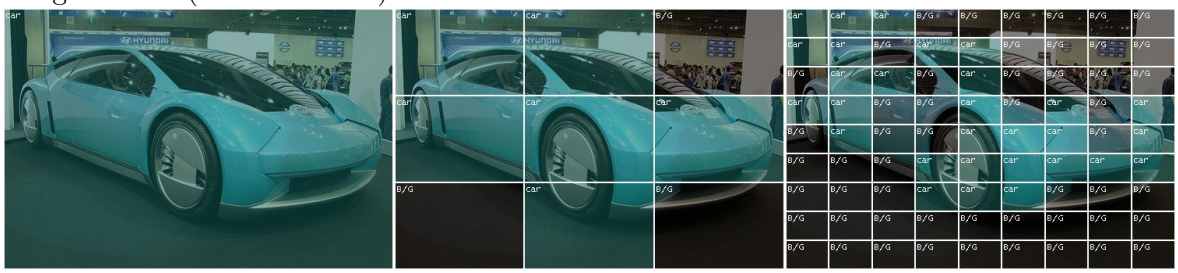


Figure 11: Some further results using 3×3 branching.

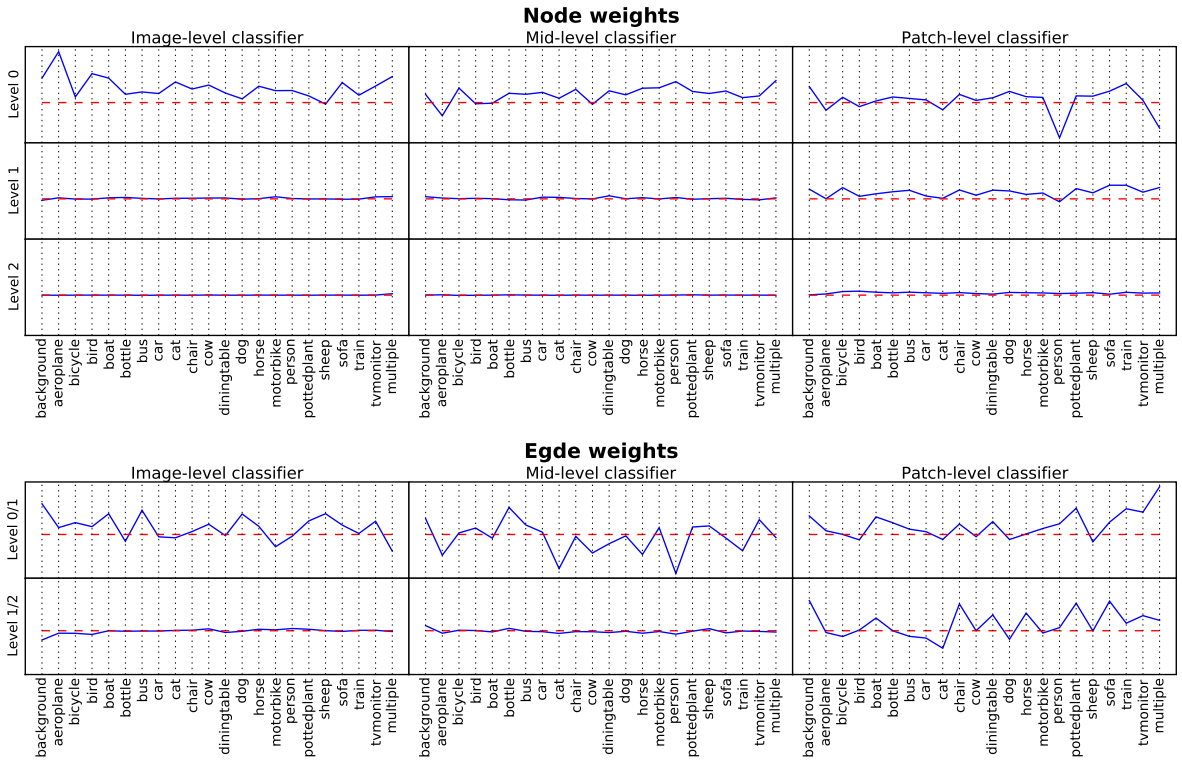


Figure 12: The complete weight-vector for our model using 3×3 branching.