Submitted to IEEE PAMI April 24 1999, revised March 2001, accepted July 2001.

Combining belief networks and neural networks for scene segmentation

Xiaojuan Feng*

Informatics Laboratory

National Institute for Biological Standards and Control

Blanche Lane, South Mimms

Potters Bar, Herts., EN6 3QG, UK

email: xfeng@nibsc.ac.uk

Christopher K. I. Williams†, Stephen N. Felderhof

Division of Informatics

The University of Edinburgh

5 Forrest Hill, Edinburgh EH1 2QL, UK

email: c.k.i.williams@ed.ac.uk, stephenf@dai.ed.ac.uk

March 16, 2001

*The work described in this paper paper was carried out while the author was at the Division of Informatics, University of Edinburgh.

†To whom correspondence should be addressed.

Abstract

We are concerned with the problem of image segmentation, in which each pixel is assigned to one of a predefined finite number of labels. In Bayesian image analysis, this requires fusing together local predictions for the class labels with a prior model of label images. Following the work of [5] we consider the use of tree-structured belief networks (TSBNs) as prior models. The parameters in the TSBN are trained using a maximum likelihood objective function with the EM algorithm, and the resulting model is evaluated by calculating how efficiently it codes label images.

A number of authors have used Gaussian mixture models to connect the label field to the image data. In this paper we compare this approach to the scaled-likelihood method of [42, 31], where local predictions of pixel classification from neural networks are fused with the TSBN prior. Our results show a higher performance is obtained with the neural networks. We evaluate the classification results obtained, and emphasise not only the *maximum a posteriori* segmentation, but also the uncertainty, as evidenced e.g. by the pixelwise posterior marginal entropies. We also investigate the use of *conditional* maximum likelihood training for the TSBN, and find that this gives rise to improved classification performance over the ML-trained TSBN.

1 Introduction

We are concerned with the problem of image segmentation in which each pixel will be assigned to one out of a finite number of classes. Our work is applied to images of outdoor scenes, with class labels such as "sky", "road", "vegetation", etc. Such scenes are typically complex, involving many different objects, and some of these objects can be highly variable (e.g. trees). This means that model-based approaches are not readily applicable.

Much work on scene segmentation has been based on approaches which first segment the whole image into regions, and then classify each region. To carry out this task successfully it is important to classify each region using not only its own attributes, but also to take account of the *context* of the other regions. Taking account of context can be handled in two ways; either by searching for a consistent interpretation of the whole scene, or by taking account of the local context in which a region finds itself [16]. Examples of the whole-scene method are [34, 29], while the local-context method is used in [50, 51]. A major problem for these approaches is that the process of region creation can be unreliable, leading to under- or over-segmentations.

An alternative approach allows the segmentation to emerge along with the classification process. This can be formulated in the Bayesian framework, using a prior model which represents our knowledge about the likely patterns of labels in the image, and a likelihood function which describes the relationship of the observations to the class labels[1]. Two main types of prior model have been investigated, called non-causal Markov random fields (MRFs) and causal MRFs in the statistical image modelling literature. In the graphical models community these two types of models are known as undirected and directed graphical models respectively [21].

Early work on Bayesian image modelling concentrated on non-causal MRFs, see e.g. [3, 13, 27]. One disadvantage of such models is that they suffer from high computational complexity, for example the problem of finding the *maximum a posteriori* (MAP) interpretation given an image is (in general) NP-hard.

The alternative causal MRF formulation uses a directed graph. The most commonly used form of these models is a tree-structured belief network (TSBN) structure, as illustrated in Figure 1. For image modelling the standard dependency structure is that of a quadtree. One attractive feature of the TSBN model is its hierarchical multiscale nature, so that long-range correlations can readily be induced. By contrast, non-causal MRFs typically have a "flat", non-hierarchical structure. Also, we shall see that inference in TSBNs can be carried out in a time linear in the number of pixels, using a sweep up the tree from the leaves to the root, and back down again. In the graphical models literature this inference procedure is known as Pearl's message-passing scheme [35]. This algorithm is also known as the upward-downward algorithm [40, 24], being a generalisation to trees of the standard Baum-Welch forward-backward algorithm for HMMs (see e.g. [37]). One disadvantage of TSBNs is that the random field is non-stationary. For example, in Figure 1, the common parent of the fourth and fifth pixels from the left in $\boldsymbol{X}^L$ is the root node, whilst the third and fourth pixels share a parent in the layer above. This can give rise to "blocky" segmentations.

TSBN models have been used by a number of authors for image analysis tasks. Bouman and Shapiro [5] introduced such a model using discrete labels in the nodes for an image segmentation task. Pérez *et al* [36] have discussed MAP and MPM (maximum posterior marginal) inference on TSBNs for image processing

[1] Some early work in this direction was carried out by Feldman and Yakimovsky [10].

tasks. Laferté *et al* [19, 20] have extended this model using a multiscale feature pyramid image decomposition and by using the EM algorithm for parameter estimation. Cheng and Bouman [6] have investigated trainable multiscale models, using decision trees to compactly represent conditional probability tables (CPTs) in their model.

TSBN models can also be used for continuously-valued Gaussian processes in one and two dimensions. These are the generalisation of Kalman filter models from chains to trees. They have been studied by a number of groups, notably Prof. Willsky's group at MIT, who have developed the theory and derived fast algorithms for problems such as optical flow estimation, surface reconstruction and texture segmentation [2, 26, 11, 25]; see also [17]. Also, Crouse *et al* [7] have used a multiscale TSBN to model wavelet coefficients, and DeBonet and Viola [9] have used an interesting tree-structured network for image synthesis using non-Gaussian densities. However, we require a prior over class-label images, and so the work on TSBNs with discrete-valued nodes is the most germane for our purposes.

As mentioned above, exact inference procedures for non-causal MRFs are, in general, NP-hard. However, we note that there has been important recent work on approximate inference procedures for such graphs, using local probability propagation schemes [46]. These local schemes (which are guaranteed to give the correct answer on graphs without loops, see [35]) give only approximate answers for the grid-structured graphs typically used in image analysis. Other work on message-passing schemes in "loopy" graphs includes the decoding of error-correcting codes [12]. The work in [46] can be criticised on the basis that "flat" (i.e. non-hierarchical) non-causal MRF models are used, although it should be possible to apply similar message passing schemes on loopy hierarchical graphs (see section 7 for further discussion).

Our work is carried out on a database of images of outdoor scenes (see Section 4 for details) for which both colour images and the corresponding label images are available. This paper makes a number of contributions:

- We investigate the effects of the adaptation (or training) of the parameters of the tree in response to data. Two methods are considered. In the first the tree is trained to maximise the probability of the labelled images; we call this maximum likelihood (ML) training. This is similar to the work of Laferté *et al* in [20], but we allow a non-stationary parameterisation of the tree which reflects regularities in the image database. In the second method the tree is trained to maximise the probability of the correct label image given the raw input image; we call this conditional maximum likelihood (CML) training.

- The quality of the ML-trained TSBN model has been evaluated by comparing how well it codes a test set label images. This performance is compared with a number of other coding schemes including the JPEG-LS lossless codec. We also provide an analysis of TSBN coding which allows us to quantify the benefits of using higher levels of the tree (which correspond to longer-range correlations).

- The TSBN comprises the prior aspect of the Bayesian model, and we also require a likelihood term whereby the image data influences the segmentation. The direct approach is to produce a generative model of the probability density of pixel features given the image class. For example in [5] the class conditional densities were modelled using Gaussian mixture models. We compare this approach with an alternative one, where neural networks are used to make local predictions given the pixel features. These predictions can then be combined with the prior in a principled manner using the scaled-likelihood method (see Section 2.2 for further details). The effects of combining these methods with the ML and CML trained trees are also investigated. We evaluate the performance of the segmentation algorithms not only using pixelwise classification rates, but also through the analysis of posterior pixelwise entropies and through the conditional probabilities of the label images given the colour images.

The remainder of this paper is organised as follows: In Section 2 we describe the TSBN model and the scaled-likelihood method in more detail, and explain how the inference can be carried out. In Section 3 we derive equations for training the TSBN using maximum likelihood (ML) and conditional maximum likelihood (CML) methods, and image coding using TSBNs. In Section 4 we give details on the data and the training of various models. In Section 5 results are presented concerning image coding and the estimation of the CPTs and in Section 6 we analyse the classification results.

2 Belief network and neural network models for image segmentation

2.1 Generative model

Our model for the data is illustrated in Figure 1. The observed data $\boldsymbol{Y}$ is assumed to have been generated from an underlying process $\boldsymbol{X}$, where $\boldsymbol{X}$ is a TSBN. The network is arranged in layers. At the highest level (level 0) there is one node $\boldsymbol{X}^0$, which has children in level 1. The lower levels are denoted $\boldsymbol{X}^1, \boldsymbol{X}^2$ and so on, down to $\boldsymbol{X}^L$ in level L. The fundamental property of belief networks is their encoding of conditional independences [35, 15]. The layered structure means that the distribution of $\boldsymbol{X}^n$, for $1 \leq n \leq L$, given all coarser scale nodes is only dependent on $\boldsymbol{X}^{n-1}$. Indeed, the tree-structure of the network means that any node in $\boldsymbol{X}^n$ is only dependent on a single node in $\boldsymbol{X}^{n-1}$. Typically in our experiments each parent node has four children, giving rise to a quadtree-type architecture.

Each node is a multinomial variable, taking on one of C class labels. These labels are those used for the segmentation, e.g. road, sky, vehicle etc[2]. The links between the nodes are defined by conditional probability tables (CPTs). As it does not have any parent, the root node has an unconditional prior distribution instead of a CPT.

The nodes in $\boldsymbol{X}^L$ have a one-to-one correspondence with the observed data $\boldsymbol{Y}$. (The observed data in our case will be features derived from blocks of pixels, rather than individual pixels in the raw image.) The model for the observations illustrated in Figure 1 is that the observation Y_i is dependent only on the corresponding variable X_i^L, i.e. $P(Y_i = y_i | \boldsymbol{X} = \boldsymbol{x}) = P(Y_i = y_i | X_i^L = x_i^L)$. The overall model for $P(\boldsymbol{X}, \boldsymbol{Y})$ is

$$P(\boldsymbol{X}, \boldsymbol{Y}) = P(\boldsymbol{X}) \prod_{i=1}^{K} P(Y_i | X_i^L). \tag{1}$$

[2] Note that it is not necessary for the hidden nodes to be C-class multinomial variables. We have used this for convenience, and because it gives rise to a simple initialisation of the TSBN-training, as described in Section 4.5.

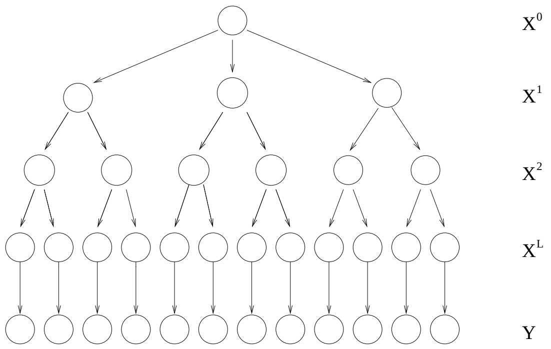


Figure 1: A 1-D graphical model illustrating a small tree-structured belief network. The layers in $\boldsymbol{X}$ are denoted $\boldsymbol{X}^0$, $\boldsymbol{X}^1$ and so on, down to $\boldsymbol{X}^L$; in this case $L = 3$. $\boldsymbol{Y}$ denotes the raw image information.

2.2 Likelihood model

Above we have described a fully generative model for $\boldsymbol{Y}$, defined by CPTs from the root downwards. Since each pixel is composed of a number of components in feature space, $P(Y_i = y_i | X_i^L = x_i^L)$ is a multivariate density function. One method is to use Gaussian mixture density, as used in [5] and described in section 4.4.

However, the database we are using provides both the raw data $\boldsymbol{Y}$ and labelled segmentations $\boldsymbol{X}^L$. Given this information it is natural to train classifiers (e.g. neural networks) to predict X_i^L given Y_i, and it is well known that neural networks trained with various error functions (including cross-entropy and mean-squared error) approximate the posterior probabilities $P(X_i^L|Y_i)$[38]. Fusion of these predictions into the belief network for $\boldsymbol{X}$ cannot then be achieved immediately, as this requires $P(Y_i|X_i^L)$ and not $P(X_i^L|Y_i)$ terms. However, by using Bayes' theorem we obtain

$$P(Y_i|X_i^L) = \frac{P(X_i^L|Y_i)P(Y_i)}{P(X_i^L)}. \tag{2}$$

For inference on $\boldsymbol{X}$, the data $\boldsymbol{Y}$ is fixed and so the factors $P(Y_i)$ need not be considered further. Defining the *scaled likelihood* $L(X_i^L, Y_i)$ for a particular location i as

$$L(X_i^L, Y_i) \stackrel{def}{=} \frac{P(Y_i|X_i^L)}{P(Y_i)} = \frac{P(X_i^L|Y_i)}{P(X_i^L)}, \tag{3}$$

we obtain

$$P(\boldsymbol{X}|\boldsymbol{Y}) \propto P(\boldsymbol{X}) \prod_{i=1}^{K} L(X_i^L, Y_i). \tag{4}$$

By replacing $L(X_i^L, Y_i)$ with $\hat{L}(X_i^L, Y_i) = \hat{P}(X_i^L|Y_i)/\hat{P}(X_i^L)$ we obtain a principled method for the fusion of the local predictions $\hat{P}(X_i^L|Y_i), i = 1, \ldots, K$ with the global prior model $P(\boldsymbol{X})$. (The notation $\hat{P}$ denotes that $\hat{P}(X_i^L|Y_i)$ and $\hat{P}(X_i^L)$ are *estimates* of the desired probabilities.) This method of combining neural networks with belief networks has been suggested (for HMMs) in Smyth[42] and Morgan and Bourlard[31]. There is an interesting connection between the scaled likelihood and the mutual information $I(X_i, Y_i)$ between the random variables X_i and Y_i, where $I(X_i, Y_i) = E_{X_i Y_i}[\log P(X_i|Y_i)/P(X_i)]$. The mutual information is the expected value of the log scaled likelihood.

As described above, we would need to have and train a separate neural network to predict $P(X_i|Y_i)$ for each pixel. This is clearly undesirable, both in terms of the computational effort and in the amount of data required. The solution we have adopted is to train just one neural network, but to make the position coefficients of the pixel be a part of the inputs to the neural network.

To use the scaled likelihood, we require not only $\hat{P}(X_i|Y_i)$ but also $\hat{P}(X_i)$. In our images it will turn out that $P(X_i)$ should depend on the position of pixel i, as we know that in the ensemble of images there are regularities such as the sky appearing at the top. There are a number of ways to approach the estimation of $P(X_i)$. One is to train a network to predict the class labels given the position of the pixel. An alternative would be to use the relationship $P(X_i) = \int P(X_i|Y_i)P(Y_i)\, dY_i$ and to approximate the integral with an average over appropriate feature vectors. In our experiments (see Section 6) we compared a spatially invariant $\hat{P}(X_i)$ to one derived from the pixelwise marginals of the ML-trained TSBN; the results obtained were very similar.

A potential advantage of the scaled-likelihood method is that the generative model for $P(Y_i|X_i)$ may be quite complex, although the predictive distribution $P(X_i|Y_i)$ is actually quite simple. This means that the generative approach may spend a lot of resources on modelling details of $P(Y_i|X_i)$ which are not particularly relevant to the task of inferring $\boldsymbol{X}$.

2.3 Inference

Given a new image $\boldsymbol{Y} = \boldsymbol{y}$ we wish to carry out inference on $\boldsymbol{X}^L$, given the probabilistic model. Computing the posterior $P(\boldsymbol{X}^L = \boldsymbol{x}^L | \boldsymbol{Y} = \boldsymbol{y})$ would be highly expensive, as it would require enumerating all possible C^K states in $\boldsymbol{X}^L$. There are two alternatives that are computationally feasible, (i) the computation of the posterior marginals $P(X_i^L = x_i^L | \boldsymbol{Y} = \boldsymbol{y})$ giving rise to a segmentation based on maximum posterior marginals (MPM) and (ii) the MAP interpretation of the data $\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x}} P(\boldsymbol{X} = \boldsymbol{x} | \boldsymbol{Y} = \boldsymbol{y})$[3]. These can be achieved by Pearl's λ-π message passing schemes, as described in [35]. These schemes are non-iterative and involve one upward and one downward pass through the tree. Details are given of the MPM computation in Appendix A, along with a method for scaling the calculation to avoid underflow, and the computation of the marginal likelihood $P(\boldsymbol{x}^L)$.

3 Training the TSBN

Above it was assumed that the parameters used to define $P(\boldsymbol{X})$ are known. In fact we estimated these from training data. Let $\boldsymbol{\theta}$ denote these parameters, which are the prior probabilities of the root node and the CPTs in the tree. Let x_{il}, $l = 1, \ldots, C$ denote the possible values of X_i, and let pa_{ik}, $k = 1, \ldots, C$ denote the set of possible values taken on by Pa_i, the parent of X_i. The parameter θ_{ikl} denotes the CPT entry $P(X_i = x_{il} | Pa_i = pa_{ik})$. Note that $\sum_l \theta_{ikl} = 1$. For simplicity the symbols X_i and Pa_i are dropped, and the probability is written as $P(x_{il} | pa_{ik})$.

For training the prior model it is assumed that a number of observation images $\boldsymbol{y}^m$ and associated labelled images $\boldsymbol{x}^{Lm}$ are available, where $m = 1, \cdots, M$ is the index to the images in the training set. Let $\boldsymbol{\phi}$ denote the parameters in the likelihood model $P(\boldsymbol{y} | \boldsymbol{x}, \boldsymbol{\phi})$.

We discuss in turn maximum likelihood training (§3.1) and conditional maximum likelihood training (§3.2).

[3] Note that finding the configuration $\boldsymbol{x}^*$ is not likely to be equivalent to finding the configuration that maximises $P(\boldsymbol{X}^L = \boldsymbol{x}^L | \boldsymbol{Y} = \boldsymbol{y})$.

3.1 Maximum likelihood

In maximum likelihood training the parameter vector, $\boldsymbol{\theta}$, is estimated so that

$$\hat{\boldsymbol{\theta}}^{ML} = \arg\max_{\boldsymbol{\theta}} \prod_{m=1}^{M} P(\boldsymbol{x}^{Lm}, \boldsymbol{y}^m | \boldsymbol{\theta}, \boldsymbol{\phi}) = \arg\max_{\boldsymbol{\theta}} \prod_{m=1}^{M} P(\boldsymbol{y}^m | \boldsymbol{x}^{Lm}, \boldsymbol{\phi}) P(\boldsymbol{x}^{Lm} | \boldsymbol{\theta}). \tag{5}$$

We can see that the likelihood model parameters $\boldsymbol{\phi}$ and the TSBN model parameters $\boldsymbol{\theta}$ can be estimated separately by choosing the likelihood model parameters to maximise $\prod_m P(\boldsymbol{y}^m|\boldsymbol{x}^{Lm}, \boldsymbol{\phi})$ and the prior model parameters to maximise $\prod_m P(\boldsymbol{x}^{Lm}|\boldsymbol{\theta})$. Assuming that the likelihood model is fixed, we obtain $\hat{\boldsymbol{\theta}}^{ML} = \arg\max_{\boldsymbol{\theta}} \prod_{m=1}^{M} P(\boldsymbol{x}^{Lm}|\boldsymbol{\theta})$. The optimisation can be carried out using the EM algorithm. It uses bottom-up and top-down message passing to infer the posterior probabilities of the hidden nodes in the E-step, and then uses the expected counts of the transitions to re-estimate the CPTs [40, 24, 21].

The re-estimation formulas can be derived directly by maximising Baum's auxiliary function $Q(\bar{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \sum_{m=1}^{M} \sum_{\boldsymbol{x}_h^m} P(\boldsymbol{x}_h^m | \boldsymbol{x}^{Lm}, \boldsymbol{\theta}) \log [\, P(\boldsymbol{x}_h^m, \boldsymbol{x}^{Lm} | \bar{\boldsymbol{\theta}}) \,]$ over $\bar{\boldsymbol{\theta}}$ (the new estimated parameter vector), where $\boldsymbol{x}_h^m$ denotes the "hidden" variables $\boldsymbol{x}^m \backslash \boldsymbol{x}^{Lm}$ on pattern m. The update for each entry in a CPT is given by

$$\bar{\theta}_{ikl} = \frac{\sum_{m=1}^{M} P(x_{il}^m, pa_{ik}^m | \boldsymbol{x}^{Lm}, \boldsymbol{\theta})}{\sum_{m=1}^{M} \sum_{l'} P(x_{il'}^m, pa_{ik}^m | \boldsymbol{x}^{Lm}, \boldsymbol{\theta})}. \tag{6}$$

The joint probability $P(x_{il}^m, pa_{ik}^m | \boldsymbol{x}^{Lm}, \boldsymbol{\theta})$ can be obtained locally using the λ-π message passing scheme (see Appendix A for details).

This update gives a separate update for each link in the tree. Given limited training data this is undesirable. If the set of variables sharing a CPT is denoted as $\boldsymbol{X}_I$, then the EM parameter update is given by

$$\tilde{\theta}_{Ikl} = \frac{\sum_{m=1}^{M} \sum_{X_i \in \boldsymbol{X}_I} P(x_{il}^m, pa_{ik}^m | \boldsymbol{x}^{Lm}, \boldsymbol{\theta})}{\sum_{m=1}^{M} \sum_{X_i \in \boldsymbol{X}_I} \sum_{l'} P(x_{il'}^m, pa_{ik}^m | \boldsymbol{x}^{Lm}, \boldsymbol{\theta})}. \tag{7}$$

If only the $\boldsymbol{Y}$ information is available, one can still carry out maximum likelihood training of the model; both parameter sets $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ would be adapted in this case. This is known as unsupervised learning, and is described for TSBNs in [20]. A disadvantage of the scaled-likelihood method is that it cannot be used for unsupervised learning as $P(\boldsymbol{Y})$ is not available.

In [5] (Section III C), it appears that the parameters are re-estimated on the *test* image, which is unusual from the standard pattern recognition methodology, where model parameters are estimated from training data and then fixed when applied to test images. We follow the standard methodology.

3.2 Conditional maximum likelihood

In the CML procedure, the objective is to predict correctly the labels $\boldsymbol{x}^L$ associated with the evidence $\boldsymbol{y}$. The parameters are then estimated by maximising the probability of the correct labelling given the evidence $\boldsymbol{y}$,

$$\hat{\boldsymbol{\theta}}^{CML} = \arg\max_{\boldsymbol{\theta}} \prod_{m=1}^{M} P(\boldsymbol{x}^{Lm}|\boldsymbol{y}^m, \boldsymbol{\theta}, \boldsymbol{\phi}) = \arg\max_{\boldsymbol{\theta}} \prod_{m=1}^{M} \frac{P(\boldsymbol{x}^{Lm}, \boldsymbol{y}^m|\boldsymbol{\theta}, \boldsymbol{\phi})}{P(\boldsymbol{y}^m|\boldsymbol{\theta}, \boldsymbol{\phi})}. \tag{8}$$

By analogy to the Boltzmann machine, we observe that computing the conditional probability requires computation of (1) the probability $P(\boldsymbol{x}^{Lm}, \boldsymbol{y}^m|\boldsymbol{\theta}, \boldsymbol{\phi})$ in the *clamped phase* (i.e. with $\boldsymbol{x}^{Lm}$ and $\boldsymbol{y}^m$ fixed), and (2) the probability $p(\boldsymbol{y}^m|\boldsymbol{\theta}, \boldsymbol{\phi})$ in the *free-running* phase (with only $\boldsymbol{y}^m$ fixed)[4].

Below we assume that the likelihood model $P(\boldsymbol{y}|\boldsymbol{x}^L, \boldsymbol{\phi})$ is fixed, so that the objective function is viewed as a function of $\boldsymbol{\theta}$ only. To carry out the optimisation in Equation 8 we take logarithms and define

$$L_f(\boldsymbol{\theta}) = \sum_{m=1}^{M} \log\ P(\boldsymbol{y}^m|\boldsymbol{\theta}, \boldsymbol{\phi}) \qquad \text{and} \qquad L_c(\boldsymbol{\theta}) = \sum_{m=1}^{M} \log\ P(\boldsymbol{x}^{Lm}, \boldsymbol{y}^m|\boldsymbol{\theta}, \boldsymbol{\phi}). \tag{9}$$

Here we have used the subscripts c and f to mean "clamped" and "free". Using the decomposition of $P(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{\phi}) = P(\boldsymbol{x}|\boldsymbol{\theta})P(\boldsymbol{y}|\boldsymbol{x}^L, \boldsymbol{\phi})$, $L_c(\boldsymbol{\theta})$ can be further simplified as

$$L_c(\boldsymbol{\theta}) = \sum_{m=1}^{M} \log[\ P(\boldsymbol{y}^m|\boldsymbol{x}^{Lm}, \boldsymbol{\phi})P(\boldsymbol{x}^{Lm}|\boldsymbol{\theta})\ ] = \sum_{m=1}^{M} \log P(\boldsymbol{y}^m|\boldsymbol{x}^{Lm}, \boldsymbol{\phi}) + \sum_{m=1}^{M} \log P(\boldsymbol{x}^{Lm}|\boldsymbol{\theta}). \tag{10}$$

Then, to find $\hat{\boldsymbol{\theta}}^{CML}$ in Equation 8 we need to maximise

$$L(\boldsymbol{\theta}) = \sum_{m=1}^{M} \log \frac{P(\boldsymbol{x}^{Lm}, \boldsymbol{y}^m|\boldsymbol{\theta}, \boldsymbol{\phi})}{P(\boldsymbol{y}^m|\boldsymbol{\theta}, \boldsymbol{\phi})} = L_c(\boldsymbol{\theta}) - L_f(\boldsymbol{\theta}). \tag{11}$$

[4]The use of the terminology clamped and free-running here follows that in [18].

Unfortunately the EM algorithm is not applicable to the CML estimation, because the CML criterion is expressed as a rational function[14] . However, maximisation of Equation 11 can be carried out in various ways based on the gradient of $L(\boldsymbol{\theta})$. In speech analysis [18, 39], methods based on gradient ascent have been used. The scaled conjugate gradient optimisation algorithm [30, 4] was used in our work. To use this search method we need to calculate the gradient of $L(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$.

Letting $n_{ikl}^m = P(x_{il}^m, pa_{ik}^m | \boldsymbol{y}^m, \boldsymbol{\theta}, \boldsymbol{\phi})$ and $m_{ikl}^m = P(x_{il}^m, pa_{ik}^m | \boldsymbol{x}^{Lm}, \boldsymbol{\theta})$, it can be shown (see Appendix B for details) that

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \theta_{ikl}} = \frac{\sum_{m=1}^{M} (m_{ikl}^m - n_{ikl}^m)}{\theta_{ikl}}, \tag{12}$$

where m_{ikl} and n_{ikl} can be obtained by propagating $\boldsymbol{y}^m$ and $\boldsymbol{x}^{Lm}$ respectively, see Equation 23.

When maximising $L(\boldsymbol{\theta})$ it must be ensured that the probability parameters remain positive and properly normalised. The softmax function is used to meet these constraints. We define $\theta_{ikl} = e^{z_{ikl}} / \sum_{l'} e^{z_{ikl'}}$ where the z_{ikl}'s are the new unconstrained auxiliary variables and θ_{ikl} always sums to one over the l index by construction. The gradients w.r.t. z_{ikl} can be expressed entirely in terms of θ_{ikl}, m_{ikl}^n and n_{ikl}^n,

$$\frac{\partial L(\boldsymbol{\theta})}{\partial z_{ikl}} = -\sum_{n=1}^{N} \left[m_{ikl}^n - n_{ikl}^n - \theta_{ikl} \sum_{l'} (m_{ikl'}^n - n_{ik'}^n) \right]. \tag{13}$$

3.3 Image coding

A TSBN provides a generative, probabilistic model of label images. We can evaluate the quality of a model for the label process by evaluating the likelihood of a test set of label images under the model. By calculating $-\log_2 P(\boldsymbol{x}^L)/(\#\text{labelled pixels in the image})$ we obtain the coding cost in bits/pixel. The minimum attainable coding cost is the entropy (in bits/pixel) of the generating process. As the computation of $P(\boldsymbol{x}^L)$ is intractable in MRF models, we compare the TSBN results to those from the lossless JPEG-LS codec [44, 45], available from `http://www.hpl.hp.com/loco/`.

3.3.1 Coding cost under a TSBN model

Using a TSBN to model the distribution of images, the marginal likelihood of a label image $\boldsymbol{x}^L$ can be calculated efficiently at the root node X^0 of the tree (see Appendix A.2).

Below we will also consider the effect of truncating the tree at a level below the root of the tree. In this case, instead of having one large tree, the image model consists of a number of smaller trees (and correlations between the different trees are then ignored). This allows us to quantify the benefits of using the higher levels in the tree, which correspond to longer-range correlations. The priors for each of the smaller trees are calculated by propagating the prior $\pi(x^0)$ through the downward CPTs to obtain a prior for each root. The likelihood of an image under the truncated model is simply the product of the likelihoods of the subimages as computed in each of the smaller trees.

4 Experimental details

4.1 Data

Colour images of out-door scenes from the Sowerby Image Database[5] of British Aerospace are used in our experiments. The database contains both urban and rural scenes. They feature a variety of every-day objects from roads, cars and houses to lanes and fields in various places near Bristol, UK. All the scenes were photographed using small-grain 35mm transparency film, under carefully controlled conditions. Each image of the database has been digitised with a calibrated scanner, generating a high quality 24-bit colour representation.

Both colour images and their corresponding label images are provided in the database. The label images were created by over-segmenting the images, and then hand labelling each region produced. There were 92 possible labels, organised in a hierarchical system. We combined labels to produce seven classes, namely “sky”, “vegetation”, “road markings”, “road surface”, “building”, “street furniture” and “mobile object”. For instance, the class “street furniture” is a combination of many types such as road sign, telegraph pole

[5]This database can be made available to other researchers. Please contact Dr. Andy Wright or Dr. Gareth Rees, Advanced Information Processing Department, Advanced Technology Centre - Sowerby, BAE SYSTEMS Ltd, PO Box 5 Filton, Bristol, BS34 7QW, UK, email: `gareth-s.rees@baesystems.com` for details.

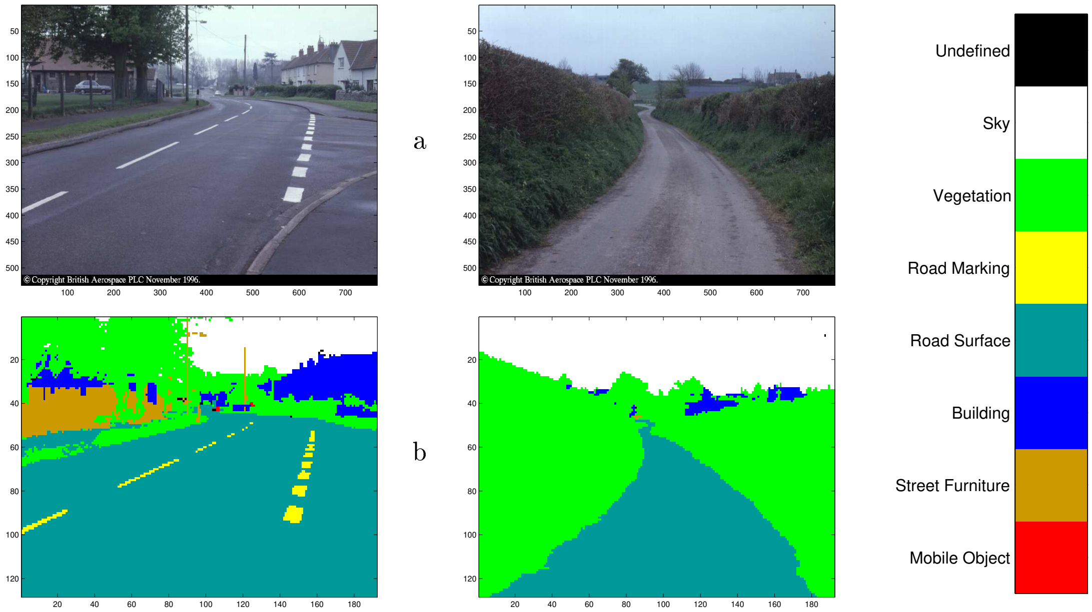


Figure 2: A rural and an urban scene and their hand-labelled classification. (a) Original images. (b) Hand-labelled classification. On the right is a key describing the labels used.

and bounding object.

Figure 2a shows two scenes from the test image set of the database, one rural and one urban. Figure 2b shows their hand-labelled classifications; different grey-levels in the label image correspond to the seven different possible classes. The original 104 images were divided randomly into independent training and test sets of size 61 and 43 respectively. The full-resolution colour images of size 512 by 768 pixels were downsampled into 128 by 192 regions of size 4 by 4 pixels. The label of the reduced region was chosen by majority vote within the region, with ties being resolved by an ordering on the label categories. From now on we will refer to the reduced label images as label images because the original label images will no longer be used.

4.2 Feature extraction

An important step in classification is that of feature selection. Initially forty features were extracted from each region. Among them, six features were based on the R, G, B colour components, i.e., the mean and variance of overall intensity in the region, the colour hue angle (sine and cosine) [1], (R-B) and (2G-R-B)/2 (as used by [34]), where R, G and B indicate the means of the red, green and blue components respectively. The texture features are the grey-level difference vectors (GLDV) textural features [48, 47] of contrast, entropy, local homogeneity, angular second moment, mean, standard deviation, cluster shade and cluster prominence. The GLDV features were extracted based on the absolute difference between pairs of gray levels at a distance $d = 1$ apart at four angles $\theta = 0°, 45°, 90°, 135°$. The (x, y) location of the region was also included in the feature space, as described in Section 2.2. Each feature was normalised by using a linear transformation to have zero mean and unit variance over the training set.

It is useful to limit the number of features used because increasing the number of features increases the free parameters which need to be optimised in neural network training phase. A generalised linear model (GLM) using the normalised features as inputs and softmax outputs [4] was used in feature selection. For each input, the sum of the absolute values of the weights coming out of that input in the trained GLM was calculated, and the twenty-one features for which this sum was larger than unity were retained. This selection procedure is based on the idea that more important features will tend to give rise to larger weights (*cf* the Automatic Relevance Determination idea of MacKay and Neal [32]).

4.3 MLP training

Multi-Layer Perceptrons (MLPs) were used for the task of predicting $P(X_i^L|Y_i)$. As explained in Section 2.2, these probabilities were estimated by a MLP that takes as input both the non-positional feature vector and position of pixel i. The retained features produced a feature vector for each region and were fed to a MLP with 21 input nodes, 7 output nodes and one hidden layer which was trained to classify each region into one of the seven classes. The activation functions of the output nodes and hidden nodes were the softmax function and tanh sigmoid functions respectively. The error function used in the training process was cross-entropy for multiple classes (see [4]). A scaled conjugate gradient algorithm was used to minimise the error function. The training is performed using over 51,000 regions extracted from the training image dataset, and validating on an independent validation dataset of over 15,000 regions. The validation dataset was used in order to choose the optimal number of hidden nodes in the MLP; eventually the best performance on the validation set is obtained for a MLP with 30 hidden nodes.

The training dataset for MLP training was formed by choosing randomly up to 150 regions for each class from each single image. By doing so we tried to use equal numbers of regions from each class in the training set of the MLP. The aim of this rebalancing of the training set is to give the net a better chance to learn about the infrequent classes (see [4], p 224). The probabilities for each class in the training set of the MLP, denoted by $\tilde{P}(c_k), k = 1, \cdots, C$, were estimated simply by evaluating the fraction of the training set data points in each class. The corresponding probabilities for all pixels in the whole of the training set images are denoted by $P(c_k), k = 1, \cdots, C$. These turned out to be $\tilde{P}(c_k) = (0.2089, 0.2122, 0.0453, 0.2122, 0.1549, 0.1140, 0.0525)$ and $P(c_k) = (0.1468, 0.3665, 0.0023, 0.4205, 0.0491, 0.0078, 0.0070)$. The ordering of the classes is "sky", "vegetation", "road markings", "road surface", "building", "street furniture" and "mobile object". These two sets of prior probabilities are very different; $\tilde{P}(c_k)$ is almost uniformly distributed over all classes, but $P(c_k)$ is biased towards classes two and four, corresponding to "vegetation" and "road surface" respectively.

Since the training set for the MLP reweights the classes according to $\hat{P}(c_k)$, it is necessary to consider what effect this will have on the scaled likelihood. In fact, following [4] (p. 223) we find that

$$P(c_k|y_i) = \frac{1}{Z} \frac{\tilde{P}(c_k|y_i)}{\tilde{P}(c_k)} P(c_k), \tag{14}$$

where y_i is the input to the network at pixel i, $\tilde{P}(c_k|y_i)$ is the network output for class k, $P(c_k|y_i)$ is the compensated network output and Z is the normalising factor used to make $\sum_k P(c_k|y_i)$ sum to one. Hence we see that the scaled likelihood $P(c_k|y_i)/P(c_k)$ is equal to $\tilde{P}(c_k|y_i)/\tilde{P}(c_k)$ up to an unimportant constant. We call $\tilde{P}(c_k|y_i)$ the raw MLP prediction, and $P(c_k|y_i)$ (as given by Equation 14) the *compensated* MLP prediction. A segmentation can be obtained by choosing the most probable class at each pixel independently. We call these the raw MLP and compensated MLP segmentations, when using the uncompensated and compensated predictions respectively.

4.4 Gaussian mixture model training

In section 4.3 we have described how MLPs were trained to relate the image features to labels. The alternative approach is to build class-conditional density estimators for each class, and to use this along with Bayes' rule to make predictions. Following [5, 20] we have used Gaussian mixture models (GMMs) for this task. Specifically the `cluster` program available from `http://www.ece.purdue.edu/~bouman` was used. The same training set was used as for the MLP. We considered three different feature sets (i) the average R, G, B values in a region, (ii) the 21 features used to train the MLP and (iii) all 40 features. In addition two different settings of the `cluster` program were used, allowing for either diagonal or full covariance matrices for the Gaussians. The program selects the number of mixture components automatically using a MDL criterion; the recommended initialisation of starting with three times as many components as features was used.

The GMMs for each class were combined with the prior probabilities for each class (the $P(c_k)$'s given in section 4.3) to produce pixelwise classifications. The overall classification accuracies were 68.72%, 49.76%, 75.95%, 71.38%, 77.45%, 71.92% for the 3-full, 3-diag, 21-full, 21-diag, 40-full, 40-diag models respectively on the test set. The GMM model with highest pixelwise performance (namely 40-full) was then used in further experiments (see section 6 for further details). The number of mixture components in the 40-full model for each of the seven classes was 9, 9, 4, 9, 8, 7, 6 respectively.

We have found that the trained GMM sometimes makes very confident misclassifications and that this can cause underflow problems when evaluating the conditional probability $P_{TSBN}(\boldsymbol{x}^{L*}|\boldsymbol{y})$, the conditional probability of the "ground truth" labelling $\boldsymbol{x}^{L*}$ given the image $\boldsymbol{y}$ (see section 6.4). For this reason we replaced the likelihood term $P(Y_i|X_i^L)$ with $\max(P(Y_i|X_i^L), \varepsilon)$, where ε was set to 10^{-29}, the minimum value needed to avoid underflow in the ML-TSBN.

4.5 TSBN training

The TSBN used was basically a quadtree, except that there were six children of the root node to take into account the 2:3 aspect ratio of the images. For a down-sampled image with a total of 128×192 pixels, we took each pixel in the down-sampled image as a leaf node in a belief network and built up an eight-level TSBN with a total of 32756 links between nodes at adjacent levels. If each link had a separate CPT, a very large training set would be needed to ensure the CPTs were well determined, and this in turn implies that huge computational resources could be needed in order to find a suitable minimum of the CML objective function. In practice such an approach is clearly impractical. One technique for dimensionality reduction in this case is to tie CPTs. In our experiments all of the CPTs on each level were constrained to be equal, except for the transition from level 0 to level 1, where each table was separate. This flexibility allows knowledge about the broad nature of scenes (e.g. sky occurs near to the top of images) to be learned by the network, as is indeed reflected in the learned CPTs (see section 5).

For training the ML-TSBN, the network parameters were initialised $\boldsymbol{\theta}$ in a number of different ways. It was found that the highest marginal likelihood in the training data was obtained when the initial values of $\boldsymbol{\theta}$ were computed using probabilities derived from downsampled version of the images. The sparse-data problem appeared in the initial values of the CPTs because some pairings do not occur in the training data[6]. We dealt with the problem by adding a small quantity η to each conditional probability $p(c_k|c_i)$ for $k = 1, \cdots C$, then normalising the modified probabilities. We have used $\eta = min_k(p(c_k|c_i))/100$ for the case that at least one $p(c_k|c_i) \neq 0$, otherwise $\eta = 1/C$. The plot of likelihood against iteration number levelled off after 30 iterations. In the database some pixels are unlabelled; assuming these values are "missing at random", we treated them as uninstantiated nodes, which can easily be handled in a belief network framework.

The CML training was initialised at the ML-TSBN solution. The plots of conditional likelihood against iteration number had levelled off after 44 iterations of CML training by scaled conjugate gradient optimisation for both the GMM and MLP predictors.

The upward (λ) propagation in the tree takes around 10s, and the downward (π) propagation around 40s on a SGI R10000 processor; the tree has over 30,000 nodes.

We have made available the C++ code for TSBN training and inference, along with a MATLAB demonstration which calls these functions at `http://www.dai.ed.ac.uk/daidb/people/homes/ckiw/code/cbn.html`.

4.6 Combining pixelwise predictions with trees

We now have GMM and MLP local predictors, and ML and CML trained TSBNs. This gives rise to a large number of possible combinations of pixelwise predictors with trees. The ones that we have investigated are

[6]The reason why it is important to consider this is that if a CPT entry is set to zero, the EM algorithm will not move it away from zero during training.

1. raw GMM pixelwise predictions
2. compensated GMM pixelwise predictions (spatially-uniform compensation)
3. compensated GMM pixelwise predictions (using marginals of ML-TSBN)
4. GMM likelihood + ML-TSBN
5. GMM likelihood + CML-TSBN
6. raw MLP pixelwise predictions
7. compensated MLP pixelwise predictions (spatially-uniform compensation)
8. compensated MLP pixelwise predictions (using marginals of ML-TSBN)
9. MLP scaled-likelihood + ML-TSBN
10. MLP scaled-likelihood + CML-TSBN

The TSBN methods calculated the scaled likelihoods as described in Section 4.3, and MAP inference was used for the pixelwise predictions. For entries 3 and 8 (compensation using the marginals of the ML-TSBN) note that different compensation probabilities are used in the six regions of the image defined by the six CPTs from the root to level 1. The performance of these methods is investigated in sections 5 and 6.

5 Results from TSBN training

In this section we describe the results from training the TSBN using both ML and CML training. We first discuss label-image coding results using the ML-trained tree, and then inspect the learned CPTs from the ML-trained TSBN and the CML-trained TSBN.

5.1 Image coding results

In this section we present results comparing the ML-trained TSBN and lossless JPEG coding. The relevant theory has been described in Section 3.3 and the details of the TSBN training are given in Section 4.5.

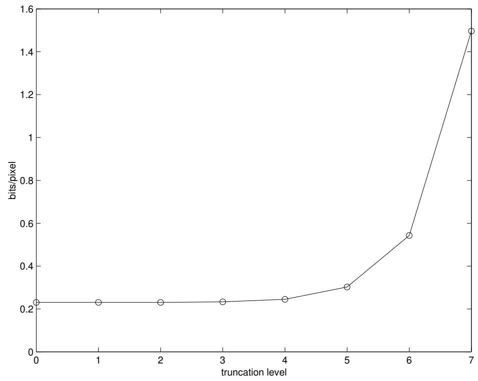


Figure 3: Bit rate (bits/pixel) as a function of the truncation level in the TSBN.

The average bit rate for the TSBN model was 0.2307 bits/pixel (bpp). For comparison purposes the JPEG-LS codec gave an average bit rate of 0.2420 bpp. We also tried compressing the label images using Lempel-Ziv coding using the Unix utility `gzip`; this gave 0.3779 bpp. The fact that a similar level of compression performance is obtained by JPEG-LS and the TSBN suggests that the TSBN is a reasonably good model of the label images.

Using the "truncated tree" scheme discussed in Section 3.3, we can analyse the TSBN results further. Figure 3 shows the bit rate (in bits/pixel) evaluated as a function of truncating the tree at levels 0 to 7. By the time level 4 has been reached (corresponding to a 8×8 block size), almost all of the benefit has been attained.

5.2 The learned CPTs

The CPTs derived using ML training are shown in Figure 4. Note that six separate CPTs were used for the transition from the root node to level 1, as explained in Section 4.5.

We can also calculate the prior marginals at each node in the tree, by simply taking the prior from the root node and passing it through the relevant CPTs on the path from the root to the node under consideration[7]. The fact that there are six CPTs for the root to level 1 transition means that there are, in effect, six different prior marginals in levels 1 to 7, defined by the 2:3 aspect ratio of the image. These prior marginals are shown in Figure 5. It may not be easy to interpret the CPTs/marginals as a permutation of the state labels at

[7]This can also be achieved using Pearl's propagation scheme outlined in Appendix A, with every leaf node uninstantiated.

a node and corresponding permutations of the incoming and outgoing CPTs would leave the overall model unchanged; however, it appears that the downsampling initialisation means that this is not a large problem.

Analysing Figures 4 and 5, we see that (1) The prior marginals at level 7 reflect the overall statistics of the images. The sky, vegetation and road surface classes are the most frequently occurring, the sky class is more likely to be found in the top half of the images and road surface in the bottom half. Similar patterns are detectable at level 1 of Figure 5, although the vegetation label is less prevalent in the upper half at this level. (2) The trained CPTs in levels 1 to 7 exhibit a strong diagonal structure, implying that the children are most likely to inherit their parent's class. (3) The level 0 to level 1 CPTs need to be read in conjunction with the root's prior distribution to provide a good explanation of the level 1 prior marginals.

Although Laferté *et al* [20] have carried out EM training of a TSBN, we note that they only estimated CPTs that were tied on a layer-by-layer basis. For our data Figures 4 and 5 show that relaxing this constraint can be useful.

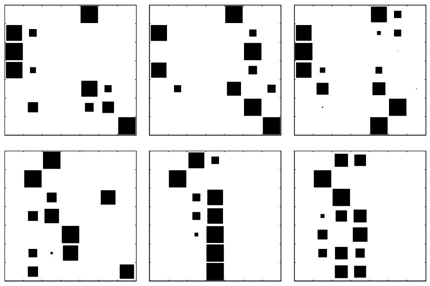
The CPTs and prior marginals obtained with CML training were similar to those shown in Figures 4 and 5 respectively. This is probably due to the fact that CML training was initialised at the ML-TSBN solution for both GMM and MLP predictors.

6 Segmentation results and performance evaluation

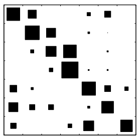
We now turn to the classification of the testing images. Often classification performance is evaluated on pixelwise accuracies. However, for complex real-world classification task, such as ours, this does not tell the whole story. There are a number of other factors that concern us, most notably the fact that we are predicting the labels of pixels in an image, and that spatial coherence is important. We also note that the fractions of pixels from different classes are tremendously different, and that the ground-truth labels used in assessing performance are not 100 percent correct because of both the downsampling process, and also because of inaccuracies in the hand-labelling process. Therefore, it is a difficult task to assess the quality of classification derived from the various methods, which may also depend on the uses the classification will be put to. An early reference to assessing the quality of segmentations is [22]. More recently, there has

(a) Prior for root node:

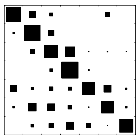
(b) From root node to Level 1:



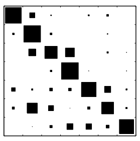
(c) From Level 1 to Level 2:



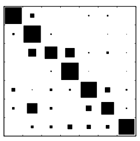
(d) From Level 2 to Level 3:



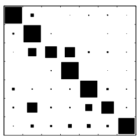
(e) From Level 3 to Level 4:



(f) From Level 4 to Level 5:



(g) From Level 5 to Level 6:



(h) From Level 6 to Level 7:

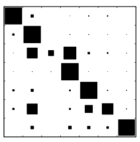


Figure 4: Estimated prior for the root and CPTs (ML training) of an eight-level belief network after being trained on training images. The area of each black square is proportional to the value of the relevant probability. (a) The prior probabilities at the root node. (b) The six independent CPTs on the links from root node to its six children on the first level. (c)-(h) CPTs for the links between adjacent levels from Level 1 to Level 7 respectively. The seven labels are 1-Sky, 2-Vegetation, 3-Road marking, 4-Road surface, 5-Building, 6-Street furniture, 7-Mobile object. The CPTs have entry (1,1) in the top left-hand corner, and are read with "from level l" indexing the rows and "to level $l+1$" indexing the columns.

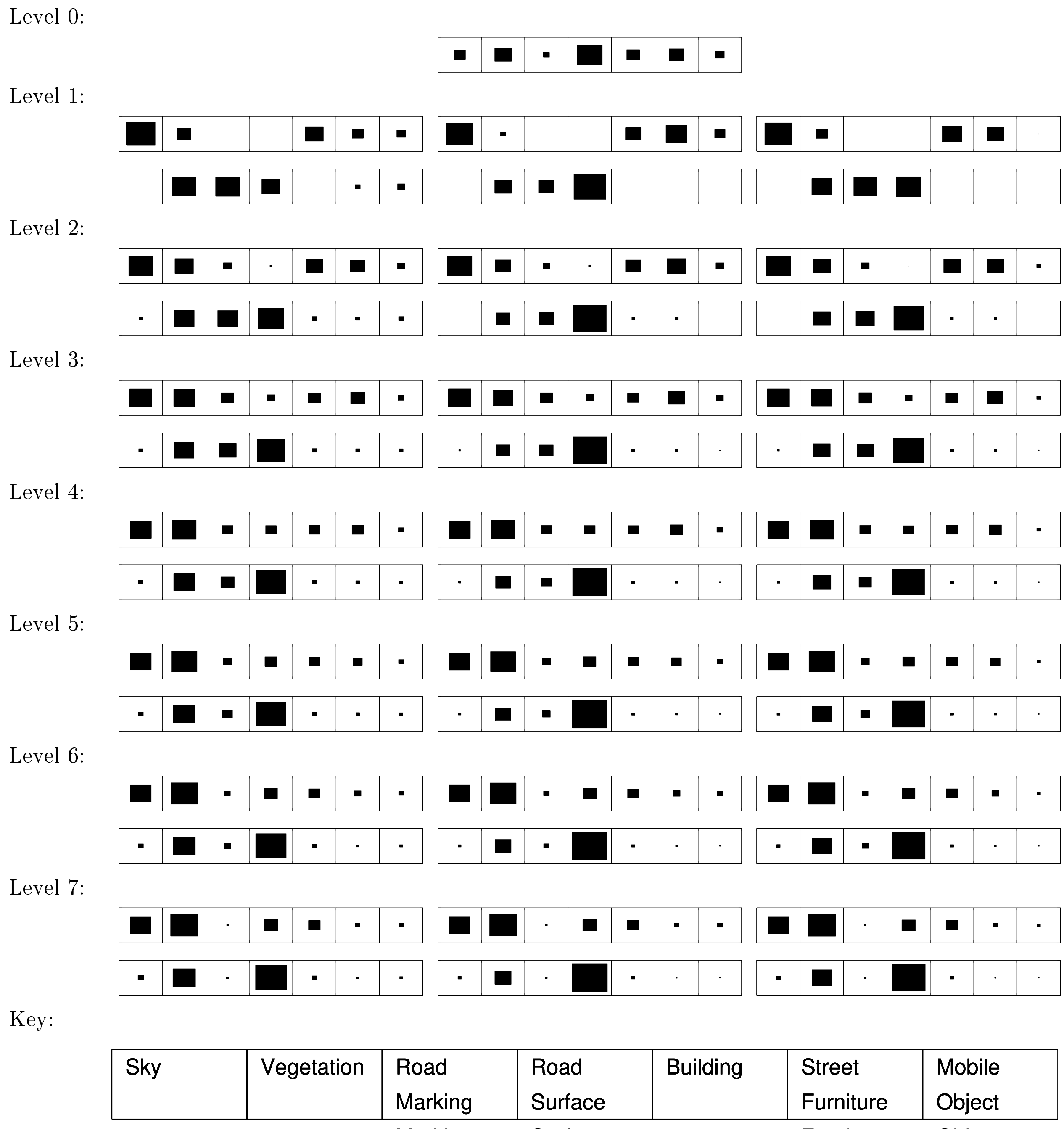


Figure 5: The prior marginals after training with the ML algorithm. The area of each black square is proportional to the value of the relevant probability. See text for further details.

been some realisation that the aim of segmentation may not be to return just a single segmentation, but multiple solutions [33], or a probability distribution over segmentations $P(\boldsymbol{x}^L|\boldsymbol{y})$. This posterior distribution can be explored in many ways; below we describe two, namely (i) posterior marginal entropies and (ii) the evaluation of the conditional probability $P(\boldsymbol{x}^{L^*}|\boldsymbol{y})$, where $\boldsymbol{x}^{L^*}$ is the "ground truth" image for given input data $\boldsymbol{y}$.

In this section we compare the performance of classification based on the smoothness of the segmented image in Section 6.1, the pixelwise prediction accuracies in Section 6.2, the marginal entropies in Section 6.3 and the conditional probability in Section 6.4.

6.1 Smoothness

For the rural scene in Figure 2, Figure 6 shows classifications using most of the combinations outlined in section 4.6. The classifications obtained from the single-pixel methods typically have a lot of high-frequency noise, due to locally ambiguous regions. Both the ML- and CML-trained trees tend to smooth out this noise.

A similar smoothing can be obtained using a majority filter [23], where one simply chooses the most common class within a window centered on the pixel of interest. However, one drawback with this is that majority-filter smoothing with a reasonably-sized window tends to remove fine detail, such as road markings; in contrast it seems that the TSBN methods yield something like an adaptive smoothing, depending on the strength of the local evidence. Also, note that majority-filtering does not return a probability distribution over segmentations.

6.2 Pixelwise classification accuracy

Table 1 shows the pixelwise classification accuracy for each class, and the overall accuracy for each of the ten methods listed in Section 4.6. For the TSBN methods the MAP segmentation result is reported; the MPM results were similar, although they were generally worse by a few tenths of one percent. The most noticeable feature is that the performance obtained with the MLP methods is superior to that from the GMM methods.

Looking at the results in detail, we notice that the raw results for both the GMM and MLP (columns 1 and 6) are improved by compensation (columns 2 and 7 resp). The compensated methods simply give more

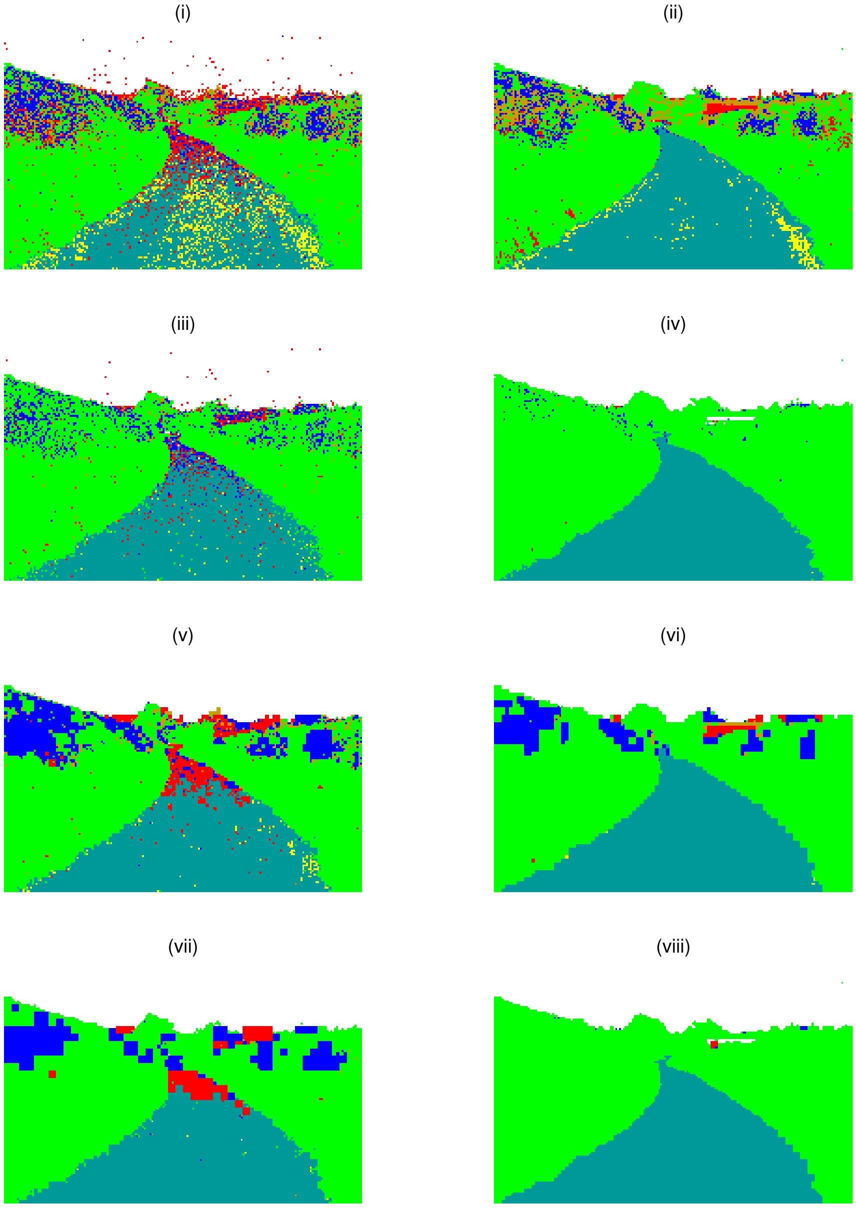


Figure 6: Classification of a rural scene. (i) raw GMM pixelwise predictions, (ii) raw MLP pixelwise predictions, (iii) compensated GMM pixelwise predictions, (iv) compensated MLP pixelwise predictions, (v) MAP segmentation for GMM + ML-TSBN, (vi) MAP segmentation for MLP + ML-TSBN, (vii) MAP segmentation for GMM + CML-TSBN, (viii) MAP segmentation for MLP + CML-TSBN.

weight to the more frequently occurring classes, as can be seen by comparing columns 1 and 2, and 6 and 7. There are only small differences between the spatially uniform compensation of columns 2 and 7, and the ML-TSBN marginal compensation scheme of columns 3 and 8.

Columns 4 and 8 combine pixelwise evidence with the ML-TSBN. For both GMM and MLP local models, it is perhaps surprising that the performance decreases as compared to columns 3 and 7 respectively. (This is a fair comparison as the methods of columns 3 and 7 use the marginals of the ML-TSBN, but not the correlation structure).

Columns 5 and 10 show the performance of the GMM and MLP local models combined with trees trained using the CML method on the relevant data. In both cases the performance is better than fusion with the ML-TSBN and for the MLP this method obtains the best overall performance.

For comparison, we note that McCauley and Engel [28] compared the performance of Bouman and Shapiro's SMAP algorithm against a pixelwise Gaussian classifier on a remote sensing task, and found that the overall classification accuracy of SMAP was a 3.6% higher (93.4% vs 89.8%).

The reasons for the superior performance of the MLP in our experiments are not entirely clear. However, we note that the test images are a relatively diverse set of images (although drawn from the same distribution as the training images). It may be that features that are important to the MLP classifier are similar in the training and test images, while other features (whose distribution has to be modelled by the GMMs) do vary between training and test sets. In contrast, some other evaluations in the literature (e.g. [28]) use only a single test image with training data drawn from a subset of the pixels; in this case the issue of inter-image variability does not arise. We also note that the comparison of the GMM and MLP classifiers was carried out using a training set of a particular size and composition; different results might be obtained if these factors were varied.

6.3 Pixel-wise entropy

We are interested in understanding the uncertainty described by $P(\boldsymbol{x}^L|\boldsymbol{y})$. It appears that the computation of the joint entropy of this conditional distribution is intractable, however posterior marginal entropies are readily computable from the posterior marginals $P(x_i^L|\boldsymbol{y})$, i.e. $H(x_i^L|\boldsymbol{y}) = -\sum_{k=1}^{C} P(x_i^L = k|\boldsymbol{y}) \log_2 P(x_i =$

Table 1: Performance of the 10 methods, showing the percentage correct for each class and overall. The second column in the table gives the overall percentage of each class in the test images.

Class	Overall percentage	1	2	3	4	5	6	7	8	9	10
sky	12.10	91.85	94.72	94.87	93.74	92.74	95.90	95.66	96.19	94.69	96.30
vegetation	40.12	61.04	78.92	77.46	67.92	75.72	79.32	92.41	90.40	81.67	90.45
road markings	0.17	55.14	42.26	44.33	43.09	27.36	78.61	68.97	67.91	70.04	67.91
road surface	39.5	62.14	78.18	80.45	70.10	73.45	94.52	97.16	96.26	94.99	96.85
building	6.11	44.19	46.98	49.67	63.70	74.92	67.69	44.43	52.73	79.40	64.60
street furniture	1.35	28.58	14.05	13.77	20.97	8.58	24.89	3.96	4.62	10.07	6.63
mobile object	0.57	58.85	43.05	43.10	72.76	76.23	49.13	28.83	32.15	78.89	44.74
overall		63.71	77.45	77.94	71.00	75.85	85.72	90.16	89.38	87.38	90.68

$k|\boldsymbol{y})$ [8]. Images displaying these posterior marginal entropies are shown in Figure 7, pertaining to the original image shown on the right in Figure 2. As expected, the pixelwise entropy is reduced by the use of the TSBN; this is particularly effective for the CML trees. Notice that the pixels which have significant posterior marginal entropy are good indicators of the pixels that are misclassified; this is especially true of the CML-TSBN + MLP combination in Figure 7. This property could well be useful information for a later stage of processing.

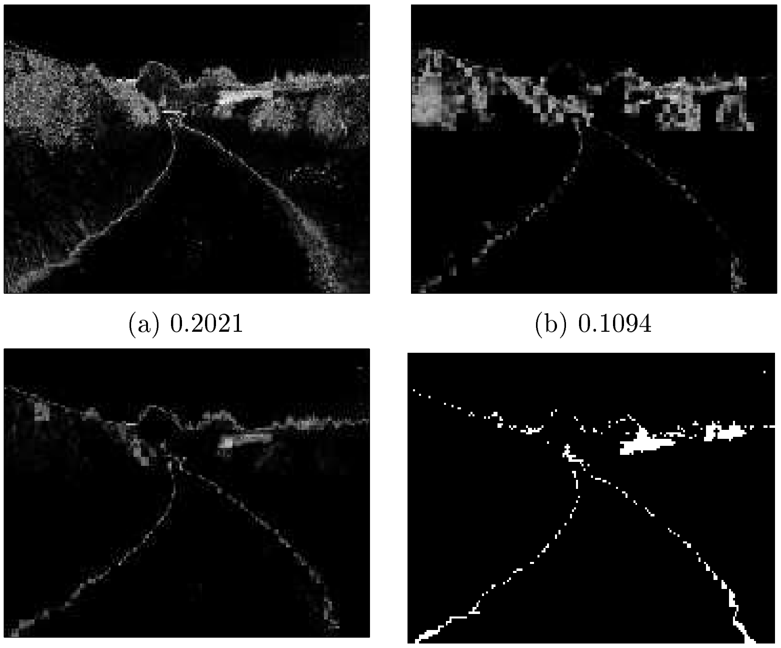
6.4 Conditional probability

If the model we have developed is a good one, then $P(\boldsymbol{x}^L|\boldsymbol{y})$ should ascribe high probability to the "ground truth" labelling $\boldsymbol{x}^{L^*}$. Different image models can be compared in terms of the relative values of $P(\boldsymbol{x}^{L^*}|\boldsymbol{y})$. In particular we compare the TSBN image models against independent pixel models.

Ignoring spatial correlations we obtain $P_{MLP}(\boldsymbol{x}^{L^*}|\boldsymbol{y}) = \prod_i P(x_i^{L^*}|y_i)$ for the MLP local prediction, and similarly for GMM local prediction. For a TSBN $P(\boldsymbol{x}^{L^*}|\boldsymbol{y})$ can be calculated as follows:

$$
\begin{aligned}
P_{TSBN}(\boldsymbol{x}^{L^*}|\boldsymbol{y}) = \frac{P(\boldsymbol{x}^{L^*},\boldsymbol{y})}{P(\boldsymbol{y})} \quad &= \quad \frac{P(\boldsymbol{x}^{L^*})\prod_i P(y_i|x_i^{L^*})}{\sum_{\boldsymbol{x}} P(\boldsymbol{x})\prod_i P(y_i|x_i^L)} & (15)\\
&= \quad \frac{P(\boldsymbol{x}^{L^*})\prod_i L(x_i^{L^*},y_i)P(y_i)}{\sum_{\boldsymbol{x}} P(\boldsymbol{x})\prod_i L(x_i^L,y_i)P(y_i)} & (16)\\
&= \quad \frac{P(\boldsymbol{x}^{L^*})\prod_i L(x_i^{L^*},y_i)}{\sum_{\boldsymbol{x}} P(\boldsymbol{x})\prod_i L(x_i^L,y_i)}. & (17)
\end{aligned}
$$

[8] During the revision of this paper we became aware that the calculation of posterior marginal entropies had been proposed independently by Pérez *et al* [36] to determine "confidence maps".



(a) 0.2021 (b) 0.1094

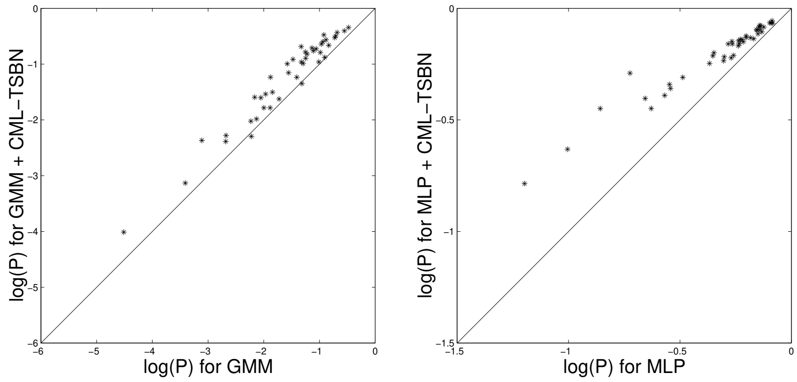
(c) 0.0487 (d)

Figure 7: Posterior marginal entropies for the MLP predictor. (a) compensated pixel-wise predictions, (b) ML-TSBN, (c) CML-TSBN. The greyscale is such that black denotes zero entropy, white denotes 2.45 bits. The number underneath each plot is the average pixelwise posterior entropy. (d) Binary image showing misclassified labels (bright) and correctly-classified labels (dark).

Equation 16 follows from Equation 15 as $P(y_i|x_i^{L^*}) = L(x_i^{L^*}, y_i)P(y_i)$. $P(\boldsymbol{x}^{L^*})$ and the denominator can be evaluated by the methods outlined in Appendix A.2.

A complexity arises in this calculation when there are pixels which do not have a label. For $P_{MLP}(\boldsymbol{x}^*|\boldsymbol{y})$, these pixels were simply ignored. For $P_{TSBN}(\boldsymbol{x}^{L^*}|\boldsymbol{y})$, the unlabelled pixels were ignored in both the numerator and denominator of Equation 17. This is achieved by setting the λ-vector at the appropriate nodes to be the vector of ones (see Appendix A for details).

In Figure 8 we plot $\frac{1}{N}\log P(\boldsymbol{x}^{L^*}|\boldsymbol{y})$ under various models. Panel (b) shows that in all 43 test images, the posterior probability under the MLP+CML-TSBN method is larger than that under the compensated MLP using an independent-pixel model. Panel (a) shows that for a similar comparison using the GMM predictor, the CML-TSBN method is better in 41 out of 43 cases. Notice also the relative scales of the plots, and especially that the GMM model makes confident mistakes on some pixels, thereby dragging down the average posterior probability.



(a) (b)

Figure 8: Comparison of $\log P(\boldsymbol{x}^{L^*}|\boldsymbol{y})/N$ for (a) compensated GMM vs GMM + CML-TSBN (b) compensated MLP vs MLP + CML-TSBN.

There are a large number of similar plots that can be made. A comparison of the posterior probabilities under the ML-trained TSBN and independent models comes up with roughly equal numbers being better coded under the two models, for both GMM and MLP predictors. The CML-trained TSBN with MLP prediction is better than all other methods on 39 out of the 43 test images; on the remaining four the ML-TSBN with MLP wins.

7 Discussion

This paper has made a number of contributions:

- We have used the EM algorithm to train the ML-TSBN and have observed that the learned parameters do reflect the underlying statistics of the training images. The quality of the probabilistic model has been evaluated in coding terms and found to be comparable to state-of-the-art methods. The "truncated tree" analysis shows over what scales correlations are important.

- We have compared the performance of GMM and MLP pixelwise classifiers on a sizable real-world image segmentation task. The performance of the discriminatively-trained MLP was found to be superior to the class-conditional GMM model. We have also shown that the scaled-likelihood method can be used to fuse the pixelwise MLP predictions with a TSBN prior.

- We have compared conditional maximum likelihood (CML) training for the tree against maximum likelihood (ML) training on a number of dimensions including classification accuracy, pixel-wise entropy and the conditional probability measure $P(\boldsymbol{x}^L|\boldsymbol{y})$.

The problem of evaluating segmentations is an old one, and a full answer may well depend on a decision-theoretic analysis which takes into account the end-use of the segmentation (e.g. for an automated driving system). However, one attractive feature of the TSBN framework is that some aspects of the posterior uncertainty can be computed efficiently, e.g. the posterior marginal entropies discussed in Section 6.3.

Clearly quadtree TSBN architectures are not the ultimate image model, as we know that run generatively they give rise to "blocky" label images. There are a number of interesting research directions which try to overcome this problem. Bouman and Shapiro [5] suggest making a more complex, cross-linked model. The problem with this is that inference now becomes much more complex (one needs to use the junction tree algorithm, see e.g. [21]). One interesting idea (suggested in [5]) is to retain Pearl-style message passing even though this is now not exact; this idea is analysed in [12] and [46]. Another approach to inference is to use alternative approximation schemes, such as the "recognition network" used in Helmholtz machines [8], or "mean-field" theory [41].

An alternative to creating a cross-linked architecture is to retain a TSBN, but to move away from a rigid quadtree architecture and allow the tree-structure to adapt to the presented image. This can be formulated in a Bayesian fashion by setting up a prior probability distribution over tree-structures. Initial results of this approach are reported in [49, 43]. We believe that the general area of creating generative models of (image) data and finding effective inference schemes for them will be a fruitful area for research.

Appendix A: Pearl's probability propagation procedures

Below we describe Pearl's scheme for probability propagation in trees, the computation of the marginal likelihood $P(\boldsymbol{x}^L|\boldsymbol{\theta})$ and a scaling procedure for this algorithm to avoid underflow.

A.1 Pearl's scheme

We first consider the calculation of the probability distribution $P(x|\boldsymbol{e})$ at node X in a TSBN, given some instantiated nodes ("evidence") $\boldsymbol{e}$.

Consider the tree fragment depicted in Figure 9 (based on Figure 4.14 in [35]). $P(x|\boldsymbol{e})$ depends on two distinct sets of evidence; evidence from the sub-tree rooted at X, denoted as $\boldsymbol{e}_x^-$, and evidence from the rest of the tree, denoted as $\boldsymbol{e}_x^+$. We shall assume that each node has a finite number of states C (each node can have a different number of states but this adds some extra notation and is not necessary in our application).

Bayes' rule, together with the independence property in the TSBNs, yields the product rule,

$$P(x|\boldsymbol{e}_x^+, \boldsymbol{e}_x^-) = \frac{P(\boldsymbol{e}_x^-|x)P(x|\boldsymbol{e}_x^+)}{P(\boldsymbol{e}_x^-|\boldsymbol{e}_x^+)} \stackrel{\text{def}}{=} \alpha(x)\lambda(x)\pi(x), \tag{18}$$

where $\lambda(x) = P(\boldsymbol{e}_x^-|x)$, $\pi(x) = P(x|\boldsymbol{e}_x^+)$ and $\alpha(x) = [P(\boldsymbol{e}_x^-|\boldsymbol{e}_x^+)]^{-1}$. Partitioning $\boldsymbol{e}_x^-$ as $\boldsymbol{e}_x^- = \cup_{i=1}^n \boldsymbol{e}_{y_i}^-$, $\lambda(x)$ is defined recursively by

$$\begin{aligned} \lambda(x) &= \prod_{i=1}^{n} \lambda_{y_i}(x) && (19) \\ \lambda_{y_i}(x) &= P(\boldsymbol{e}_{y_i}^-|x) = \sum_{k=1}^{C} P(y_{ik}|x)\lambda(y_{ik}) && (20) \end{aligned}$$

where we have assumed that node X has n children and y_{ik} is the kth value of node Y_i. $\lambda_{y_i}(x)$ is known as the λ-message sent to node X from its child node Y_i.

$\pi(x)$ is given recursively by

$$\begin{aligned} \pi(x) &= \sum_{k=1}^{C} P(x|z_k)\pi_x(z_k) && (21) \\ \pi_x(z) &= P(z|\boldsymbol{e}_x^+) = \beta\ P(z|\boldsymbol{e}_z^+) \prod_{z' \in s(X)} P(\boldsymbol{e}_{z'}^-|z) = \beta\ \pi(z) \prod_{z' \in s(X)} \lambda_{z'}(z) && (22) \end{aligned}$$

where z_k is the kth state of node Z, and $s(X)$ denotes the siblings of X (i.e. the children of Z excluding X itself) and β is the normalising factor so that the values of $\pi_x(z)$ sum to 1. In fact $\beta^{-1} = P(\boldsymbol{e}_{s(X)}^-|\boldsymbol{e}_z^+)$, where $\boldsymbol{e}_{s(X)}^-$ denotes the evidence below the siblings of X. $\pi_x(z)$ is known as the π-message sent to node X

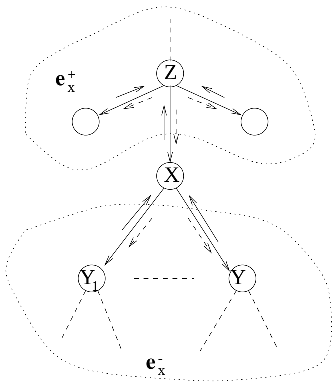


Figure 9: Fragment of causal network showing incoming message (named λ-message, shown as solid arrows) and outgoing message (π-message, broken arrows) at node X.

from its parent Z.

The propagation procedure is completed by defining the boundary conditions at the root and leaves of the tree. The π-vector at the root of the tree is equal to the prior probabilities for each of the classes. At the leaves of the tree, the λ-vector is the vector of ones if the node is uninstantiated, and equal to the vector with a single entry of 1 (and all other entries 0) corresponding to the instantiated state. The computation of $P(x|\boldsymbol{e})$ at each node in the tree can now be performed using an upward phase of λ-message passing, and a downward phase of π-message passing.

To find the *maximum a posteriori* configuration of the hidden variables $\boldsymbol{X}$ given evidence $\boldsymbol{e}$, we can use a similar message passing scheme, as described in Section 5.3 of [35].

The posterior marginal required for the EM updates and CML derivatives is given by

$$P(x_{il}^m, pa_{ik}^m | \boldsymbol{x}^{Lm}, \boldsymbol{\theta}) = \frac{1}{\sum_{k'} \pi(pa_{ik'}^m)\lambda(pa_{ik'}^m)} \; \lambda(x_{il}^m) \; \theta_{ikl} \; \pi(pa_{ik}^m) \prod_{y \in s(X_i)} \lambda_y(pa_{ik}^m), \tag{23}$$

where $s(X_i)$ is the set of nodes that are siblings of node X_i and $\lambda_y(.)$ denotes the λ-message sent to node Pa_i by node y. To show this, we partition $\boldsymbol{e}_{pa_i}^-$ as $\boldsymbol{e}_{s(X_i)}^- \cup \boldsymbol{e}_{x_i}^-$ and first calculate $P(x_{il}, pa_{ik}, \boldsymbol{e}_{s(X_i)}^-, \boldsymbol{e}_{x_i}^- | \boldsymbol{e}_{pa_i}^+)$. Using the conditional independences described by the tree we obtain

$$\begin{aligned} P(x_{il}, pa_{ik}, \boldsymbol{e}_{s(X_i)}^-, \boldsymbol{e}_{x_i}^- | \boldsymbol{e}_{pa_i}^+) &= P(pa_{ik}|\boldsymbol{e}_{pa_i}^+) P(x_{il}|pa_{ik}) P(\boldsymbol{e}_{x_i}^-|x_{il}) P(\boldsymbol{e}_{s(X_i)}^-|pa_{ik}) && (24) \\ &= \pi(pa_{ik})\theta_{ikl}\lambda(x_{il}) \prod_{y \in s(X_i)} \lambda_y(pa_{ik}). && (25) \end{aligned}$$

$P(x_{il}, pa_{ik}|\boldsymbol{e}^-_{s(X_i)}, \boldsymbol{e}^-_{x_i}, \boldsymbol{e}^+_{pa_i})$ is then computed by dividing through both sides of the equation by $P(\boldsymbol{e}^-_{s(X_i)}, \boldsymbol{e}^-_{x_i}|\boldsymbol{e}^+_{pa_i}) = P(\boldsymbol{e}^-_{pa_i}|\boldsymbol{e}^+_{pa_i})$, which can be calculated as $\sum_{k'} \pi(pa_{ik'})\lambda(pa_{ik'})$.

A.2: Marginal likelihood

Now we consider the procedure for computing $P(\boldsymbol{e}|\boldsymbol{\theta})$. Assuming X_0 is the root node, we have,

$$P(\boldsymbol{e}|\boldsymbol{\theta}) \;=\; \sum_{x_0} P(\boldsymbol{e}^+_{x_0}, x_0, \boldsymbol{e}^-_{x_0}|\boldsymbol{\theta}) = P(\boldsymbol{e}^+_{x_0}|\boldsymbol{\theta}) \sum_{x_0} P(x_0|\boldsymbol{e}^+_{x_0}, \boldsymbol{\theta}) P(\boldsymbol{e}^-_{x_0}|x_0, \boldsymbol{\theta}) = \sum_{x_0} \pi(x_0)\lambda(x_0), \tag{26}$$

where we have used $P(\boldsymbol{e}^+_{x_0}|\boldsymbol{\theta}) = 1$ since x_0 is the root node and $\boldsymbol{e}^+_{x_0}$ is empty.

A.3 Scaling Pearl's probability propagation

In order to understand where and why scaling is required for implementing the message propagation, we consider the two distinct message passing schemes separately.

Firstly, consider the definition of $\pi(x)$ in Equation 21. $\pi(x)$ is the probability of $X = x$ given the evidence $\boldsymbol{e}^+_x$, this gives $\sum_k^C \pi(x_k) = 1$ as long as the normalising factor β is applied in each time of the calculation of π-message. In this case, no scaling is needed for $\pi(x)$.

We now consider the definition of the $\lambda(x)$ in Equation 19 and $\lambda_{y_i}(x)$ of Equation 20. λ-values of node X are the product of all λ-messages sent to it by its children. Each child node forms a weighted sum of its λ-values to form the λ-message sent to its parent. The element-wise multiplication of the λ-messages in Equation 19 and the weighted sum calculation in Equation 20 cause the numerical values of the λ-vectors to decrease exponentially with distance from the leaves of the tree.

We scale $\lambda(x)$ with three goals: (1) keeping the scaled $\lambda(x)$ within the dynamic range of the computer for all nodes in the tree, (2) maintaining the local propagation mechanisms of Pearl's probability propagation, (3) recovering the true values at the end of the computation.

This is achieved by the recursive formulae

$$\hat{\lambda}(x) \;=\; \prod_{i=1}^{n} \sum_{j=1}^{C} P(y_{ik}|x)\overline{\lambda}(y_{ik}), \;\; \overline{\lambda}(x) = d_x \hat{\lambda}(x) \tag{27}$$

where $Y_1, \ldots, Y_n$ are the children of X. These equations are initialised with $\hat{\lambda}(x) = \lambda(x)$ at the leaves. d_x can be any value that gives a reasonable scaling; $d_x = 1/\sum_x \hat{\lambda}(x)$ was used in our work. The unscaled value $\lambda(x)$ is computed using $\lambda(x) = D_x^{-1}\overline{\lambda}(x)$, where D_x is the product of the scaling coefficients in the subtree rooted at X (and including d_x).

In fact if we are only interested in the calculation of $P(x|\boldsymbol{e})$, then it is not necessary to worry unduly about scaling factors; the λ-vector can simply be rescaled at each node as required, and $P(x|\boldsymbol{e})$ can be calculated from the scaled λ-vector and the π-vector by requiring that $P(x|\boldsymbol{e})$ sums to 1. However, scaling is important if we wish to calculate the marginal likelihood $P(\boldsymbol{e}|\boldsymbol{\theta})$ [9]. Referring back to Equation 26, we find that

$$\sum_{x_0} \bar{\lambda}(x_0)\pi(x_0) = D_{X_0} \sum_{x_0} \lambda(x_0)\pi(x_0) = D_{X_0} P(\boldsymbol{e}|\boldsymbol{\theta}), \tag{28}$$

where D_{X_0} is the product of all of the scaling factors used in the propagation procedure. Since D_{X_0} could be out of the machine dynamic range we compute $\log P(\boldsymbol{e}|\boldsymbol{\theta}) = \log\left[\sum_{x_0} \bar{\lambda}(x_0)\pi(x_0)\right] - \log D_{X_0}$.

Appendix B: Calculation of derivatives for CML optimisation

In this appendix we calculate the gradient of $L(\boldsymbol{\theta}) = L_c(\boldsymbol{\theta}) - L_f(\boldsymbol{\theta})$ w.r.t. $\boldsymbol{\theta}$. As in section 3.2 we suppress the dependence of $P(\boldsymbol{y}|\boldsymbol{x})$ on $\boldsymbol{\phi}$ for notational convenience.

First note that $P(\boldsymbol{y}^m|\boldsymbol{\theta})$ can be written as $P(\boldsymbol{y}^m|\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}|\boldsymbol{\theta})P(\boldsymbol{y}^m|\boldsymbol{x})$, where the sum is over all possible values of $\boldsymbol{x}$ in the TSBN. Using the conditional independence relations, $P(\boldsymbol{x}|\boldsymbol{\theta})$ is easily decomposed into a product of the transition probabilities on all links.

Following the ideas in Krogh[18] for HMMs, the derivative of $L_f(\boldsymbol{\theta})$ w.r.t θ_{ikl} is

$$\frac{\partial L_f(\boldsymbol{\theta})}{\partial \theta_{ikl}} = \sum_{m=1}^{M} \frac{1}{P(\boldsymbol{y}^m|\boldsymbol{\theta})} \frac{\partial P(\boldsymbol{y}^m|\boldsymbol{\theta})}{\partial \theta_{ikl}} \tag{29}$$

$$= \sum_{m=1}^{M} \sum_{\boldsymbol{x}^m} \frac{1}{P(\boldsymbol{y}^m|\boldsymbol{\theta})} \frac{\partial P(\boldsymbol{y}^m, \boldsymbol{x}^m|\boldsymbol{\theta})}{\partial \theta_{ikl}} \tag{30}$$

[9] Scaling issues are discussed in Pérez *et al* [36]. It appears that they addressed the issue of scaling for the computation of posterior marginals in their paper but not explicitly scaling for the computation of $P(\boldsymbol{e}|\boldsymbol{\theta})$.

$$= \sum_{m=1}^{M} \sum_{\boldsymbol{x}^m} \frac{1}{P(\boldsymbol{y}^m|\boldsymbol{\theta})} \frac{P(\boldsymbol{y}^m, \boldsymbol{x}^m|\boldsymbol{\theta})}{\theta_{ikl}} \delta(x_i^m, l)\delta(pa_i^m, k) \tag{31}$$

$$= \sum_{m=1}^{M} \sum_{\boldsymbol{x}^m} \frac{P(\boldsymbol{x}^m|\boldsymbol{y}^m, \boldsymbol{\theta})}{\theta_{ikl}} \delta(x_i^m, l)\delta(pa_i^m, k) \tag{32}$$

$$= \sum_{m=1}^{M} \frac{P(x_{il}^m, pa_{ik}^m|\boldsymbol{y}^m, \boldsymbol{\theta})}{\theta_{ikl}} \stackrel{\text{def}}{=} \frac{\sum_{m=1}^{M} n_{ikl}^m}{\theta_{ikl}}. \tag{33}$$

The step from equation 30 to equation 31 is derived from the fact that θ_{ikl} will only appear in the product $P(\boldsymbol{y}^m, \boldsymbol{x}^m|\boldsymbol{\theta})$ if X_i^m is in state l and Pa_i^m is in state k.

The derivative of the other term, $L_c(\boldsymbol{\theta})$, can be calculated in a similar manner, except that the summation over variables in the tree is now only taken over the hidden variables $\boldsymbol{x}_h = \boldsymbol{x} \backslash \boldsymbol{x}^L$. If $m_{ikl}^m = P(x_{il}^m, pa_{ik}^m|\boldsymbol{x}^{Lm}, \boldsymbol{\theta})$, then we obtain $\partial L_c(\boldsymbol{\theta})/\partial\theta_{ikl} = \sum_{m=1}^{M} m_{ikl}^m/\theta_{ikl}$.

Acknowledgements

This work is funded by EPSRC grant GR/L03088, *Combining Spatially Distributed Predictions From Neural Networks* and EPSRC grant GR/L78181 *Probabilistic Models for Sequences.* The authors gratefully acknowledge the assistance of British Aerospace in the project and in making the Sowerby Image Database available to us. We also thank Dr. Andy Wright of BAe for helpful discussions, Dr. Ian Nabney for help with NETLAB routines for neural networks, Dr. Gareth Rees (BAe) for discussions on segmentation metrics, Dr. John Elgy for introducing us to the work of [5] and Prof. Kevin Bowyer for pointing out the work of [33]. We also thank the three anonymous referees and the Associate Editor Prof. Charles Bouman for helpful comments and advice which have considerably improved the manuscript.

References

[1] D. H. Ballard and C. Brown. *Computer Vision.* Prentice-Hall, 1982.

[2] M. Basseville, A. Benveniste, K. C. Chou, S. A. Golden, R. Nikoukhah, and A. S. Willsky. Modelling and estimation of multiresolution stochastic processes. *IEEE Transactions on Information Theory,* 38:766–784, 1992.

[3] J. Besag. On the statistical analysis of dirty pirtures. *Journal of Royal Statistics, Soc. B*, 48(3):259–302, 1974.

[4] C. M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[5] C. A. Bouman and M. Shapiro. A Multiscale Random Field Model for Bayesian Image Segmentation. *IEEE Transactions on Image Processing*, 3(2):162–177, 1994.

[6] H. Cheng and C. A. Bouman. Trainable Context Model for Multiscale Segmentation. In *Proc. ICIP '98*, volume 1, pages 610–614, 1998.

[7] M. Crouse, R. Nowak, and R. Baraniuk. Wavelet-based statistical signal proccessing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46:886–902, 1998.

[8] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The Helmholtz Machine. *Neural Computation*, 7(5):889–904, 1995.

[9] J. S. de Bonet and P. A. Viola. A Non-Parametric Multi-Scale Statistical Model for Natural Images. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 773–779. MIT Press, Cambridge, MA, 1998.

[10] J. A. Feldman and Y. Yakimovsky. Decision Theory and Artificial Intelligence: I. A Semantics-Based Region Analyzer. *Artificial Intelligence*, 5:349–371, 1974.

[11] P. Fieguth, W. Karl, and A. S. Willsky. Efficient multiresolution counterparts to variational methods for surface reconstruction. *Computer Vision and Image Understanding*, 70(2):157–176, 1998.

[12] B. J. Frey and D. J. C. MacKay. A Revolution: Belief Propagation in Graphs with Cycles. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10.* MIT Press, 1998.

[13] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Trans. Pattern Analysis and Machine Intellligence*, 6(6):721–741, 1984.

[14] P. S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo. An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory*, **37**(1):107–113, 1991.

[15] F. V. Jensen. *An Introduction to Bayesian Networks*. UCL Press, London, 1996.

[16] J. Kittler. Statistical pattern recognition in image analysis. In K. V. Mardia, editor, *Statistics and Images 2*, pages 61–75. Carfax Publishing Co., 1994.

[17] S. Krishnamachari and R. Chellappa. Multiresolution Gauss-Markov random field models for texture segmentation. *IEEE Transactions on Image Processing*, 6(2):251–267, 1997.

[18] A. Krogh. Hidden markov models for labeled sequences. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, pages 140–144. IEEE Computer Society Press, 1994.

[19] J-M. Laferté, F. Heitz, P. Pérez, and E. Fabre. Hierarchical statistical models for the fusion of multiresolution data. In *Proceedings of the Fifth International Conference on Computer Vision*. IEEE, 1995.

[20] J-M. Laferté, P. Pérez, and F. Heitz. Discrete Markov Image Modelling and Inference on the Quadtree. *IEEE Transactions on Image Processing*, 9(3):390–404, 2000.

[21] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[22] M. D. Levine and A. M. Nazif. Dynamic measurement of computer generated image segmentations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(2):155–164, 1985.

[23] Lillesand, T. M. and Kiefer, R. W. *Remote Sensing and Image Interpretation*. Wiley, New York, third edition, 1994.

[24] H. Lucke. Bayesian Belief Networks as a tool for stochastic parsing. *Speech Communication*, 16:89–118, 1995.

[25] M. R. Luettgen, W. Karl, and A. S. Willsky. Efficient multiscale regularization with applications to the computation of optical flow. *IEEE Trans. Image Processing*, 3:41–64, 1994.

[26] M. R. Luettgen and A. S. Willsky. Likelihood Calculation for a Class of Multiscale Stochastic Models, with Application to Texture Discrimination. *IEEE Trans. Image Processing*, 4(2):194–207, 1995.

[27] K. V. Mardia, A. J. Baczkowski, X. Feng, and T. J. Hainsworth. Statistical methods for automatic interpretation of digitally scanned finger prints. *Pattern Recognition Letters*, 18:1197–1203, 1997.

[28] J. D. McCauley and B. A. Engel. Comparison of Scene Segmentations: SMAP, ECHO and Maximum Likelihood. *IEEE Trans. on Geoscience and Remote Sensing*, 33(6):1313–1316, 1995.

[29] W. A. McKeown, D. M. Harvey and J. McDermott. Rule-based interpretation of aerial imagery. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(5):570–585, 1985.

[30] M. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

[31] N. Morgan and H. A. Bourlard. Neural Networks for Statistical Recognition of Continuous Speech. *Proceedings of the IEEE*, 83(5):742–770, 1995.

[32] Neal, R. M. *Bayesian Learning for Neural Networks.* Springer, New York, 1996. Lecture Notes in Statistics 118.

[33] K. M. Nickels and S. Hutchinson. Textured image segmentation: returning multiple solutions. *Image and Vision Computing*, 15:781–795, 1997.

[34] Yuichi Ohta. *Knowledge-based Interpretation of Outdoor Natural Colour Scenes.* Pitman Publishing Limited, London, 1985.

[35] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[36] P. Pérez, A. Chardin, and J-M. Laferté. Noniterative manipulation of discrete energy-based models for image analysis. *Pattern Recognition*, 33(4):573–586, 2000.

[37] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of IEEE*, 77(2):257–285, 1989.

[38] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate bayesian a-posteriori probabilities. *Neural Computation*, 3 (4):461–483, 1991.

[39] S. K. Riis and A. Krogh. Hidden neural networks: a framework for HMM/NN hybrids. In *Proceedings ICASSP-97, April 21-24, Munich, Germany*, 1997.

[40] O. Ronen, J. R. Rohlicek, and M. Ostendorf. Parameter Estimation of Dependence Tree Models Using the EM ALgorithm. *IEEE Signal Processing Letters*, 2(8):157–159, 1995.

[41] L. Saul and M. I. Jordan. A mean field learning algorithm for unsupervised neural networks. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 541–554. Kluwer Academic, 1998.

[42] P. Smyth. Hidden Markov models for fault detection in dynamic systems. *Pattern Recogntion*, 27(1):149–164, 1994.

[43] A. Storkey and C. K. I. Williams. Dynamic positional trees for structural image analysis. In T. Jaakkola and T. Richardson, editors, *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pages 298–304. Morgan Kaufmann, 2001.

[44] M. Weinberger, G. Seroussi, and G. Sapiro. "LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm. In *Proc. IEEE Data Compression Conference*, 1996. Snowbird, Utah.

[45] M. Weinberger, G. Seroussi, and G. Sapiro. The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS, 1999. Hewlett-Packard Laboratories Technical Report No. HPL-98-193R1, November 1998, revised October 1999. Available from `http://www.hpl.hp.com/loco/`. To appear in *IEEE Trans. Image Processing.*

[46] Y. Weiss and W. T. Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12.* MIT Press, Cambridge, MA, 2000.

[47] R. M. Welch, M. S. Navar, and S. K. Sengupta. The effect of spatial resolution upon texture-based cloud field classifications. *Journal of Geophysical Research*, **94**:14767–14781, 1989.

[48] J. S. Weszka, C. R. Dyer, and A. Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-6**(4):269–285, 1976.

[49] C. K. I. Williams and N. J. Adams. DTs: Dynamic Trees. In M. J. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*. MIT Press, 1999.

[50] W. A. Wright. Image labelling with a neural network. In *Proceedings* 5^{th} *Alvey Vision Conference*, pages 227–232. Sheffield University Press, 1989.

[51] W. A. Wright, W. P. J. Mackeown, and P. Greenway. The use of neural networks for region labelling and scene un derstanding. In J. G Taylor, editor, *Neural Networks*, pages 165–192. Alfred Waller, 1995.