Decision Tree Fields

Sebastian Nowozin
Microsoft Research
Cambridge, UK
Sebastian.Nowozin@microsoft.com

Carsten Rother
Microsoft Research
Cambridge, UK
carrot@microsoft.com

Shai Bagon
Weizmann Institute
shai.bagon@weizmann.ac.il

Toby Sharp
Microsoft Research
Cambridge, UK
toby.sharp@microsoft.com

Bangpeng Yao
Stanford University
Stanford, CA, USA
bangpeng@cs.stanford.edu

Pushmeet Kohli
Microsoft Research
Cambridge, UK
pkohli@microsoft.com

Abstract

This paper introduces a new formulation for discrete image labeling tasks, the Decision Tree Field (DTF), that combines and generalizes random forests and conditional random fields (CRF) which have been widely used in computer vision. In a typical CRF model the unary potentials are derived from sophisticated random forest or boosting based classifiers, however, the pairwise potentials are assumed to (1) have a simple parametric form with a pre-specified and fixed dependence on the image data, and (2) to be defined on the basis of a small and fixed neighborhood. In contrast, in DTF, local interactions between multiple variables are determined by means of decision trees evaluated on the image data, allowing the interactions to be adapted to the image content. This results in powerful graphical models which are able to represent complex label structure. Our key technical contribution is to show that the DTF model can be trained efficiently and jointly using a convex approximate likelihood function, enabling us to learn over a million free model parameters. We show experimentally that for applications which have a rich and complex label structure, our model achieves excellent results.

1. Introduction

The last decade has seen the meteoric rise in the use of random field models in computer vision. Random fields have been used to model many problems including foreground-background (fg-bg) segmentation [4, 5], semantic segmentation [13, 29], and a number of other computer vision problems [31]. Many of these problems can be cast as an image labeling problem, where we are given an image $\boldsymbol{x}$ and need to predict labels $\boldsymbol{y}$. Random fields provide a way of factorizing the joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ or the posterior distribution $p(\boldsymbol{y}|\boldsymbol{x})$ into a product of local interactions.

In the classic Markov random field (MRF) we obtain the posterior distribution $p(\boldsymbol{y}|\boldsymbol{x})$ by integrating a per-pixel likelihood functions with pairwise consistency potentials ensuring a smooth solution [9, 19]. One major advance in the field was to make these smoothness cost dependent on the local image structure [5], conditioning parts of the model on the input data. In the last decade, these *conditional* random field (CRF) models [17, 30, 13] have become popular for their improved ability to capture the relationship between labels and the image.

A lot of research effort has been devoted at the development of efficient algorithms for estimating the Maximum a Posteriori (MAP) solution of such models [31, 15], and the same is true for algorithms for probabilistic inference [36, 14]. The problem of *parameter estimation* in these structured models has likewise been addressed [35, 30, 32]. However, despite these rapid developments, (most) state-of-the-art random field CRF models continue to suffer from the following limitations: (1) they are defined on the basis of a fixed neighborhood structure (except the work of [16, 23]), and (2) the potentials are assumed to have a simple parametric form with a pre-specified and fixed dependence on the image data. While it is relatively easy to think of various ways to overcome these limitations, the key research challenge is to suggest a model for which efficient and high-quality training is still tractable.

This paper introduces a new graphical model, the Decision Tree Field (DTF), which overcomes the above-mentioned limitations of existing models. We take a simple yet radical view: every interaction in our model depends on the image, and further, the dependence is non-parametric. It is easy to see that even representing such a model is extremely challenging, since there are numerous ways of defining a mapping between the image and the parameters of a unary or pairwise interaction in the graphical model.

Our model uses *decision trees* to map the image content to interaction values. Every node of every decision tree in our model is associated with a set of parameters, which are used to define the potential functions in the graphical model.

When making predictions on a novel test instance, the leaf node of the decision tree determines the effective weights.

There are a number of important reasons for the choice of decision trees to specify the dependence between potentials and image content. Firstly, decision trees are non-parametric and can represent rich functional relationships if sufficient training data is available. Secondly, the training of decision trees is scalable, both in the training set size and in that the approach can be parallelized; recent advances even allow training on graphics processors [26]. Since for most computer vision applications it is well known that the key to obtaining high predictive performance is the amount of training data, many recent works use decision trees, or a related variant of it (random Forests [6], extremely randomized trees [10], semantic texton forest [28]). In our context, decision trees give another big advantage: they allow us to efficiently and jointly learn all parameters in the model. We achieve this by using a log-concave pseudo-likelihood objective function, which is known to work well given enough training data because it is a consistent estimator [14].

Our Contributions

(1) To the best of our knowledge, we propose the first graphical model for image labelling problems which allows all potential functions to have an arbitrary dependence on the image data.

(2) We show how the dependence between potential functions and image data can be expressed via decision trees.

(3) We show how the training of the DTF model, which involves learning of a large number of parameters, can be performed efficiently.

(4) We empirically demonstrate that DTFs are superior to existing models such as random forest and common MRFs for applications with complex label interactions and large neighborhood structures.

2. Related Work

There has been relatively little work on learning image-dependent potential functions, i.e. the "conditional part" of a random field. Most algorithms for learning the parameters of a random field try to learn a class-to-class energy table that does not depend on the image content [1, 2, 20, 32, 33]. However, there have been few attempts at learning the parameters of conditional potentials [8, 22, 12]. Recently, Gould *et al.* [12] used a multiclass logistic regression classifier on a set of manually selected features, such as the length and orientation of region boundaries to obtain an image-dependent learned model for pairwise interactions. Even more recently, Cho *et al.* [8] proposed a model for image restoration whose interactions were dependent on the semantic meaning of the local image content as predicted by a classifier. Unlike our work, all the above-mentioned models either target specific tasks, or assume a particular form for the dependence of the potentials on the image content. Neither of the above-mentioned approaches is able to learn a dependency model with thousands or even millions of parameters which our model can achieve.

Decision trees are popularly used to model unary interactions, *e.g.* [27]; but with two exceptions they have not been used for pairwise or higher-order interactions. The first exception is the paper of Glesner and Koller [11], where decision trees are used to model conditional probability tables over many discrete variables in a Bayesian network. The difference is that in [11] the decisions in the tree are evaluated on states of random variables, whereas in our work we evaluate the image content and thus require no change to the inference procedure.

The second exception is the "random forest random field" [21]. Despite the similarity in name, the approach is fundamentally different from ours. Instead of defining an explicit model as we do in (2), Payet and Todorovic [21] define the model distribution implicitly as the equilibrium distribution of a learned Metropolis-Hastings Markov chain. The Metropolis-Hastings ratio is estimated by classification trees. This is a clever idea but it has a number of limitations, i) at test-time there is no choice between different inference methods but one is bound to using inefficient Markov Chain Monte Carlo (MCMC); in [21] superpixel graphs of few hundred regions are used and inference takes 30 seconds despite using advanced Swendsen-Wang cuts, and ii) the model remains *implicit*, so that inspecting the learned interactions as we will do in Section 5.3 is not possible.

In a broader view, our model has a richer representation of complex label structure. Deep architectures, such as [18] and latent variable CRFs, as in [25], have the same goal, but use hidden variables representing the presence of larger entities such as object parts. While these models are successful at representing structure, they are generally difficult to train because their negative log-likelihood function is no longer convex. In contrast, by learning powerful non-parametric conditional interactions we achieve a similar expressive power but retain convexity of the training problem.

3. Model

We now describe the details of our model. Throughout we will refer to $\boldsymbol{x} \in \mathcal{X}$ as a given observed image from the set of all possible images $\mathcal{X}$. Our goal is to infer a discrete labeling $\boldsymbol{y} \in \mathcal{Y}$, where the labeling is per-pixel, *i.e.* we have $\boldsymbol{y} = (y_i)_{i \in \mathcal{V}}$, $y_i \in \mathcal{L}$, where all variables have the same label set $\mathcal{L}$. We describe the relationship between $\boldsymbol{x}$ and $\boldsymbol{y}$ by means of an *energy function* E that decomposes into a sum of energy functions E_{t_F} over *factors* F, where F defines a subsets of variables. For example, for a pairwise factor it is $|F| = 2$. We have

$$E(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{w}) = \sum_{F \in \mathcal{F}} E_{t_F}(y_F, x_F, w_{t_F}). \tag{1}$$

By y_F we denote the collection $(y_i)_{i\in F}$, and likewise we write x_F to denote the parts of $\boldsymbol{x}$ contained in F. While there may be many different subsets in $\mathcal{F}$, we assume they are of few distinct *types* and denote the type of the factor F by t_F. The function E_{t_F} is the same for all factors of that type, but the variables and image content it acts upon differs. Furthermore, the function is *parametrized* by means of a weight vector w_{t_F} to be discussed below.

A visualization of a small factor graph model is shown in Figure 2. It has three pairwise factor types (red, blue, and green) and two unary factor types (black and turquoise). All factors depend on the image data $\boldsymbol{x}$. Figure 3 shows the "unrolled" factor graph for an image of size 4-by-3 pixels, where the basic model structure is repeated around each pixel $i \in \mathcal{V}$, and pairwise factors which reach outside the image range are omitted. In total we have $|\mathcal{F}| = 43$ factors.

The energy function (1) defines a conditional probability distribution $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w})$ as

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{w})}\exp(-E(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w})), \qquad (2)$$

where $Z(\boldsymbol{x},\boldsymbol{w}) = \sum_{\boldsymbol{y}\in\mathcal{Y}}\exp(-E(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w}))$ is the normalizing constant. So far, our model is in the general form of a *conditional random field* [17]. We now show how to use decision trees for representing E_{t_F} in (1).

With each function E_t we associate one decision tree. To evaluate $E_{t_F}(y_F, x_F, w_{t_F})$, we start at the root of the tree, and perform a sequence of tests s on the image content x_F, traversing the tree to the left or right. This process is illustrated in Figure 1. When a leaf node has been reached, we collect the *path* of traversed nodes from the root node to the leaf node. With each node q of the tree we associate a table of energy values $w_{t_F}(q, y_F)$. Depending on the number of variables y_F this energy function acts on, the table can be a vector (unary), a matrix (pairwise), or general k-dimensional array (higher order). We *sum* all the tables along the path taken and compute the energy as

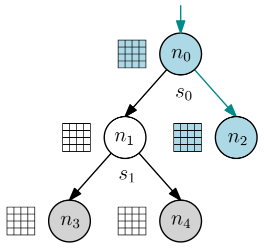

Figure 1. Summation of all energy tables along the path of visited decision nodes (shaded blue).

$$E_{t_F}(y_F, x_F, w_{t_F}) = \sum_{q\in \text{Path}(x_F)} w_{t_F}(q, y_F),$$

where $\text{Path}(x_F)$ denotes the set of nodes taken during evaluating the tree. By using one set of weights at each node we can regularize the nodes at the root of the tree to exert a stronger influence, affecting a large number of leaves; at test-time we can precompute the summation along each root-to-leaf path and store the result at each leaf.

To compute the overall energy (1) we evaluate E_{t_F} for all factors $F \in \mathcal{F}$. Although the type t_F might be the same, the

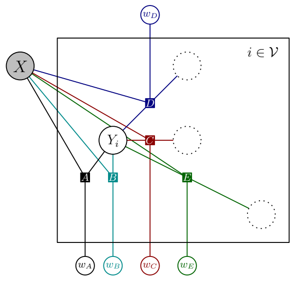

Figure 2. Neighborhood structure around each pixel with five different factor types.

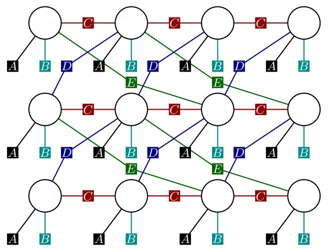

Figure 3. Unrolled factor graph (image size 4-by-3 pixels), dependencies on $\boldsymbol{x}$ and $\boldsymbol{w}$ are not shown.

function E_{t_F} depends on x_F through the evaluation of the decision tree. This allows image-dependent unary, pairwise, and higher-order interactions. The set $\mathcal{F}$ is determined by repeating the same local neighborhood structure for each pixel, as shown in Figures 2 and 3.

In summary, our model consists of a set of factor types. Each factor type contains, i) the number k of variables it acts on and their relative offsets, ii) a single decision tree, and iii) for each node in the decision tree, a table of energies of size $\mathcal{L}^k$. Given a new image $\boldsymbol{x}$, for each possible labeling $\boldsymbol{y}$ we can evaluate $E(\boldsymbol{y},\boldsymbol{x},\boldsymbol{w})$ by the above procedure.

3.1. Relation to other Models

The proposed DTF generalizes a number of popular existing image labeling methods. If we ignore pairwise and higher-order interactions in (1), then the variables are independent and making predictions for each pixel is the same as evaluating a random forests, as used in e.g. [28, 34]. Interestingly, as we will show in the experiments, even in this setting we still slightly outperform standard random forests since we learn the weights in each decision node instead of using empirical histograms; this novel modification improves predictive performance without any test-time overhead compared to random forests. For pairwise interactions we generalize simple CRFs with contrast-sensitive pairwise potentials such as in [5, 4], the *GrabCut system* [24], and TextonBoost [29]. Finally, if for the pairwise interactions we use decision trees of depth one, such that these interactions do not depend on the image content, then our model becomes a classic Markov random field prior [19].

4. Learning Decision Tree Fields

Learning the model involves selecting the neighborhood structure, the decision trees, and the weights stored in the decision nodes. During learning we are given an iid set $\{(\boldsymbol{x}_m, \boldsymbol{y}^*_m)\}_{m=1,\dots,M}$ of images $\boldsymbol{x}_m$ and ground truth labelings $\boldsymbol{y}^*_m$. Our goal is to estimate the parameters $\boldsymbol{w}$ of our model such as to predict $\boldsymbol{y}^*_m$ for a given $\boldsymbol{x}_m$. For simplifying the derivation of the learning method, we can treat the given set of images as if it would be one large collection of pixels as is done in [30].

4.1. Maximum Likelihood Learning

For learning the parameters of our model, we need to elaborate on how the parameters $\boldsymbol{w}$ define the energy. One important observation is that for a fixed set of decision trees the energy function (1) can be represented such that it is linear in the parameters $\boldsymbol{w}$. To see this, consider a single $E_{t_F}(y_F, x_F, w_{t_F})$ function and define a binary indicator function

$$B_{t_F}(q, z; y_F, x_F) = \begin{cases} 1 & \text{if } q \in \text{Path}(x_F) \text{ and } z = y_F, \\ 0 & \text{otherwise.} \end{cases}$$

Then, we can write the energy $E_{t_F}(y_F, x_F, w_{t_F})$ equivalently as a function linear in w_{t_F},

$$\sum_{q \in \text{Tree}(t_F)} \sum_{z \in \mathcal{Y}_F} w_{t_F}(q, z) B_{t_F}(q, z; y_F, x_F). \qquad (3)$$

The use of decision trees allows us to represent non-linear functions on $\boldsymbol{x}$. Although non-linear in $\boldsymbol{x}$, by the representation (3) we can parametrize this function linearly in w_{t_F}. Then, from (3) we see that the gradient has a simple form, $\nabla_{w_{t_F}(q,z)} E_{t_F}(y_F, x_F, w_{t_F}) = B_{t_F}(q, z; y_F, x_F)$.

Because (1) is linear in $\boldsymbol{w}$, the log-likelihood of (2) is a concave and differentiable function in $\boldsymbol{w}$ [14, Corollary 20.2]. This means that if computing $Z(\boldsymbol{x}, \boldsymbol{w})$ and the marginal distributions $p(y_F|\boldsymbol{x}, \boldsymbol{w})$ for all $F \in \mathcal{F}$ would be tractable, then learning the parameters by maximum likelihood becomes a convex optimization problem.

We now show how to use efficient approximate likelihood methods to learn all parameters associated to the decision trees from training data. For now we assume we are given a fixed set of factor types, including decision trees, but have to learn the weights/energies associated to the nodes of the trees. We will discuss how to learn trees later.

4.2. Pseudolikelihood

The pseudolikelihood [3] defines a surrogate likelihood function that is maximized. In contrast to the true likelihood function computing the pseudolikelihood is tractable and very efficient. The pseudolikelihood is derived from the per-variable conditional distributions $p(y_i|y^*_{\mathcal{V}\setminus\{i\}}, \boldsymbol{x}, \boldsymbol{w})$. By defining $\ell_i(\boldsymbol{w}) = -\log p(y_i|y^*_{\mathcal{V}\setminus\{i\}}, \boldsymbol{x}, \boldsymbol{w})$ we can write the regularized negative log-pseudolikelihood $\ell_{npl}(\boldsymbol{w})$ as the average ℓ_i over all pixels,

$$\ell_{npl}(\boldsymbol{w}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \ell_i(\boldsymbol{w}) - \frac{1}{|\mathcal{V}|} \sum_{t} \log p_t(w_t), \qquad (4)$$

where $p_t(w_t)$ is a *prior distribution* over w_t used to regularize the weights. We will use multivariate Normal distributions $\mathcal{N}(0, \sigma_t I)$, so that $-\log p_t(w_t)$ is of the form $\frac{1}{2\sigma_t^2}\|w_t\|^2 + C_t(\sigma_t)$ and the constant $C_t(\sigma_t)$ can be omitted during optimization because it does not depend on $\boldsymbol{w}$. For each factor type t the prior hyperparameter $\sigma_t > 0$ controls the overall influence of the factor and we need to select a suitable value by means of a model selection procedure such as cross validation.

Function (4) is convex, differentiable, and tractably computable. For optimizing (4) we use the L-BFGS numerical optimization method [37]. To use L-BFGS we need to iteratively compute $\ell_i(\boldsymbol{w})$ and the gradient $\nabla_{w_t}\ell_i(\boldsymbol{w})$. The computation of $\ell_i(\boldsymbol{w})$ and $\nabla_{w_t}\ell_i(\boldsymbol{w})$ is straightforward and yields the expressions shown in Figure 4, where we use $M(i)$ to denote the subset of $\mathcal{F}$ that involves variable y_i, and $M_t(i)$ likewise but restricted to factors of matching type, *i.e.* $M_t(i) = \{F \in M(i) : t_F = t\}$. By summing (5) and (6) over all pixels in all images, we obtain the objective and its gradient, respectively. When initializing the weights to zero we have approximately $\|\nabla_{\boldsymbol{w}}\ell_{npl}(\boldsymbol{w})\| \approx 1$. During optimization we stop when $\|\nabla_{\boldsymbol{w}}\ell_{npl}(\boldsymbol{w})\| \leq 10^{-4}$, which is the case after around 100-250 L-BFGS iterations, even for models with over a million parameters.

4.3. Learning the Tree Structure

Ideally, we would like to learn the neighborhood structure and decision trees jointly with their weights using a single objective function. However, whereas the weights are continuous, the set of decision trees is a large combinatorial set. We therefore propose to use a simple two-step heuristic to determine the decision tree structure: we learn the classification tree using the training samples and the *information gain* splitting criterion. This greedy tree construction is popular and known to work well on image labeling problems [28]. The key parameters are the maximum depth of the tree, the minimum number of samples required to keep growing the tree, and the type and number of split features used. As these settings differ from application to application, we describe them in the experimental section. Unlike in a normal classification tree, we store weights at every decision node and initialize them to zero, instead of storing histograms over classes at the leaf nodes only.

The above procedure is easily understood for unary interactions, but now show that it can be extended in a straightforward manner to learn decision trees for pairwise factors as well. To this end, if we have a pairwise factor we consider the product set $\mathcal{L} \times \mathcal{L}$ of labels and treat each label pair $(l_1, l_2) \in \mathcal{L} \times \mathcal{L}$ as a single class. Each training pair of labels becomes a single class in the product set. Given a set of such training instances we learn a classification tree over $|\mathcal{L}|^2$ classes using the information gain criterion. Instead of storing class histograms we now store weight tables with one entry per element in $\mathcal{L} \times \mathcal{L}$. The procedure extends to higher-order factors in a straightforward way.

Once the trees are obtained, we set all their weights to zero and optimize (4). During optimization the interaction between different decision trees is taken into account. This

$$\ell_i(\boldsymbol{w}) = \sum_{F\in M(i)} E_F(y_F^*, \boldsymbol{x}, w_{t_F}) + \log \sum_{y_i\in\mathcal{Y}_i} \exp\Big(- \sum_{F\in M(i)} E_F(y_i, y^*_{\mathcal{V}\setminus\{i\}}, \boldsymbol{x}, w_{t_F})\Big), \quad (5)$$

$$\nabla_{w_t}\ell_i(\boldsymbol{w}) = \sum_{F\in M_t(i)} \nabla_{w_t} E_F(\boldsymbol{y}^*, \boldsymbol{x}, w_t) - \mathbb{E}_{y_i\sim p(y_i|y^*_{\mathcal{V}\setminus\{i\}},\boldsymbol{x},\boldsymbol{w})}\Big[\sum_{F\in M_t(i)} \nabla_{w_t} E_F(y_i, y^*_{\mathcal{V}\setminus\{i\}}, \boldsymbol{x}, w_t)\Big]. \quad (6)$$

Figure 4. Objective and gradient expressions around a single variable $i \in \mathcal{V}$ for minimizing the negative log-pseudolikelihood.

is important because the tree structures are determined independently and if we were to optimize their weight independently as well, then we would suffer from overcounting labels during training. The same overcounting problem would occur if we would want to use the class histograms at the leaf nodes directly, for example by taking the negative log-probability as an energy.

4.4. Complexity of Training

The complexity to compute the overall objective (4) and its gradient is $O(|\mathcal{V}| \cdot |\mathcal{L}| \cdot N)$, where $\mathcal{V}$ is the set of pixels in the training set, $\mathcal{L}$ is the label set, and N is the number of factors in the neighborhood structure. Note that this is linear in all quantities, and independent of the order of the factors. This is possible only because of the pseudolikelihood approximation. Moreover, it is even more efficient than performing a single sweep of message passing in loopy belief propagation, which has complexity $O(|\mathcal{V}| \cdot |\mathcal{L}|^k \cdot N)$ for factors of order $k \geq 2$.

4.5. Making Training Efficient

Training a graphical model on millions of pixels is computationally challenging. We have two principled methods to make training efficient.

First, observe that our training procedure parallelizes in every step: we train the classification trees in parallel [26]. Likewise, evaluating (4) and its gradient is a large summation of independent terms, which we again compute in parallel with no communication overhead.

The second observation is that every step in our training procedure can be carried out on a subsampled training set. For classification trees we can process a subset of pixels, as in [28]. Less obvious, we can do the same thing when optimizing our objective (4). The first term in equation (4) takes the form of an empirical expectation $\mathbb{E}_{i\sim\mathcal{U}(\mathcal{V})}[\ell_i(\boldsymbol{w})]$ that can be approximated both deterministically or by means of stochastic approximation. We use a deterministic approximation by selecting a fixed subset $\mathcal{V}' \subset \mathcal{V}$ and evaluating $\ell'_{npl}(\boldsymbol{w}) = \frac{1}{|\mathcal{V}'|}\sum_{i\in\mathcal{V}'} \ell_i(\boldsymbol{w}) - \frac{1}{|\mathcal{V}'|}\sum_t \log p_t(w_t)$. We select $\mathcal{V}'$ to be large enough so this computation remains efficient; typically $\mathcal{V}'$ has a few million elements.[1]

[1]When sampling $\mathcal{V}'$ uniformly at random with replacement from $\mathcal{V}$, the *law of large numbers* guarantees the asymptotic correctness of this approximation.

4.6. Inference

We use different inference methods during test-time. For making maximum posterior marginal predictions (MPM) we use an efficient Gibbs sampler. Because the Gibbs sampling updates use the same quantities as used for computing (5) we do not have to unroll the graph. For obtaining approximate MAP predictions, we use the Gibbs sampler with simulated annealing (SA), again exploiting the model structure. Both the Gibbs sampler and the SA minimization is explained in the supplementary materials. To have a baseline comparison, we also minimize (1) using tree-reweighted message passing (TRW) by unrolling the factor graph and using the implementation of [15].

5. Experiments

We considered a broad range of applications and report experiments for three data sets. One more experiment is reported in the supplementary materials. The aim is to show that the DTF enables improved performance in challenging tasks, where a large number of interactions and parameters need to be considered and these cannot be manually tuned. Moreover, we show that conditional pairwise interactions better represent the data and lead to improved performance. As the three datasets are quite diverse, they also show the broad applicability of our system.

5.1. Conditional Interactions: Snake Dataset

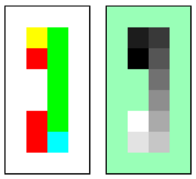
Figure 5. Input (left), labelling (right).

In this experiment we construct a task that has only very weak local evidence for any particular label and structural information needs to be propagated at test-time in order to make correct predictions. Moreover, this structure is not given but needs to be learned from training data. Consider Figure 5 to the right, illustrating the task. A "snake" shown on the input image is a sequence of adjacent pixels, and the color in the input image encodes the direction of the next pixel: red means "go north", yellow means "go east", blue means "go west", and green means "go south". Once a background pixel is reached, the snake ends. Each snake is ten pixels long, and each pixel is assigned its own label, starting from the head (black) to the tail (white), with the background taking its own label (green). Knowing about these rules, the labelling (Figure 5, right) can be perfectly reconstructed. Here, however, these rules need to be learned from training instances. Of course, in a real system the unary interactions typically provide strong cues [29, 2], but we believe that the task distills the limitations of noisy unary interactions: in this task, for making perfect predictions, the unary would need to learn about all possible snakes of length ten, of which there are very many.

We use a standard 4-neighborhood for both the MRF and the DTF models. The unary decision trees are allowed to look at every pixel in the input image, and therefore could remember the entire training image. For experimental details, please see the supplementary materials. We use a training set of 200 images, and a test set of 100 images.

The results obtained are shown in Table 1 and Figure 6. Here random forests (RF), trained unary potentials (Unary), and the the learned Markov random field (MRF) perform equally well, at around 91%. Upon examining this performance further, we discovered that while the head and tail labels are labelled with perfect accuracy, towards the middle segments of the snakes the labelling error is highest, see Table 1. This is plausible, as for these labels the local evidence is weakest. When using conditional pairwise interactions the performance improves to an almost perfect 99.4%. This again makes sense because the pairwise conditional interactions are allowed to peek at the color-codes at their neighbors for determining the directionality of the snake.

	RF	Unary	MRF	DTF
Accuracy	90.3	90.9	91.9	**99.4**
Accuracy (tail)	100	100	100	100
Accuracy (mid)	28	28	38	95

Table 1. Test set accuracies for the snake data set.

The predictions are illustrated for a single test instance in Figure 6. We see that only the DTF makes a perfect prediction. To show the uncertainty of the unary model, we visualize two samples from the model.

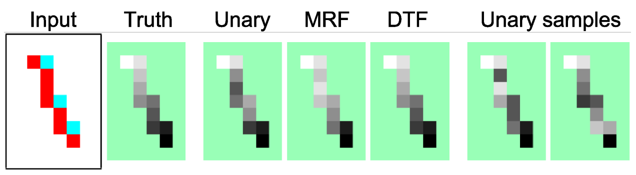

Figure 6. Predictions on a novel test instance.

5.2. Learning Calligraphy: Chinese Characters

In the previous experiment we have used a standard 4-connected neighborhood structure. In this experiment we show that by using larger conditional neighborhoods we are able to represent shape. We use the KAIST Hanja2 database of handwritten Chinese characters. We occlude each character by grey box centered on the image, but with random width and height. For more details, please see the supplementary materials. This is shown in the left-most column of Figure 7. We consider two datasets, one where we have a "small occlusion" and one with a "large occlusion" box. Note that most characters in the test set have never been observed in the training set, but a model that has learned about shape structure of Chinese characters can still find plausible completions of the input image. To this end we use one unary factor with a decision tree of depth 15. Additionally, we use a dense pairwise neighborhood structure of 8-connected neighbors at one and two pixels distance, plus a sparse set of 27 neighbors at $\{(-9,0),(-9,3),(-9,6),(-9,9),(-6,0),\ldots,(9,9)\}$. Therefore, each variable has $2\cdot(24+4+4)=64$ neighboring variables in the model. For the pairwise decision trees we use trees of depth one (MRF), or six (DTF).

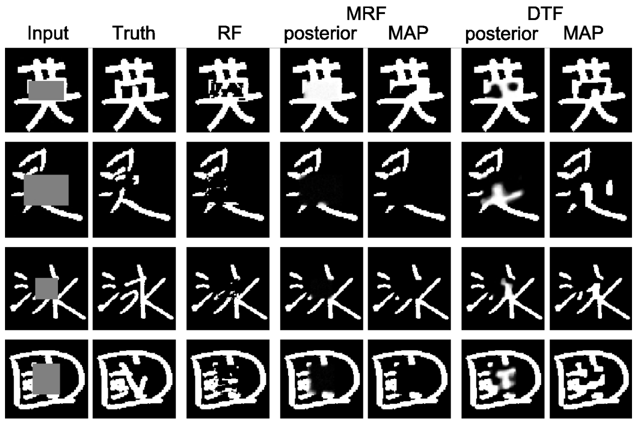

Figure 7. Test set predictions for the large occlusion case.

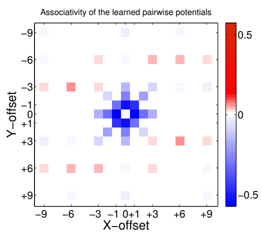

Figure 8. Pairwise associativity strength. Please see text.

The results for the large occlusion task are shown in Figure 7. Qualitatively, they show the difference between a rich connectivity structure and conditional interactions. Observe, for example, that the MRF essentially performs only a smoothing of the results while respecting local stroke-width constraints, as apparent from the MRF MAP prediction in the first row of Figure 7. In contrast, the DTF predictions hallucinate meaningful structure that may be quite different from the ground truth but bears similarity to Chinese characters. Note that we achieve this rich structure without the use of any latent variables. Because this task is an inpainting task, the quantitative assessment is more difficult since the task is truly ambiguous. We therefore report accuracies only for the small-occlusion case, where a reasonable reconstruction of the ground truth seems more feasible. We measure the per-pixel accuracy in the occluded area on the test set. For the random forest baseline we obtain 67.74%. The MRF with dense neighborhood improves this to 75.18% and the DTF obtains 76.01%.

As an example of what structure the model is able to learn, consider the visualization of the MRF pairwise inter-

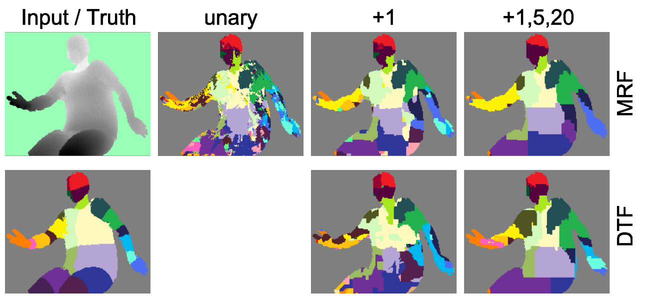

Figure 9. Test set recognition results on the training set of 30 images. We show MRF (top) and DTF (bottom) results.

actions shown in Figure 8. The figure shows for each pairwise interaction the sum of learned diagonal energies minus the sum of cross-diagonal entries. If this value is negative (shown in blue) the interaction is encouraging the pixels to take the same value. Red marks interactions that encourage pixels to take different values. The plot shows that there is a strong local smoothing term, but interestingly it is not symmetric. This can be explained by the fact that horizontal strokes in Chinese characters are typically slanted slightly upwards [7]. Note that we discovered these regularities automatically from the training data.

5.3. Accurate body-part detection

We consider the task of body part classification from depth images, as recently proposed in [27]. Given a 2D depth image, and a foreground mask, the task is to label each pixel as belonging to one of 31 different body parts, as shown in Figure 9. Despite the variations in pose and body sizes [27] obtains high-quality recognition results by evaluating a random forest for each pixel, testing local and global depth disparities. In this task, the label set has a large amount of structure, but it is not clear that a sufficiently complex unary classifier, when given the image, cannot implicitly represent this structure reasonably well. Here we show that by adding pairwise interactions we in fact improve the recognition accuracy. Moreover, once we make the interactions conditional, accuracy further improves.

The experimental setup is as follows. We use two subsets of the annotated data of [27] for training: 30 depth images, and 1500 depth images. In both cases we use a fixed separate set of 150 depth images for testing. We train 4 unary decision trees for all models. For the pairwise models, we use the following neighborhood sizes, i) "+1" for adding a 4-neighborhood one pixel away, ii) "+5" for an 8-neighborhood five pixels away, and iii) "+20" when adding an 8-neighborhood twenty pixels away. In the "+1,5,20" configuration, each variable has 4+8+8=20 neighbors. For each of the pairwise interactions we train two trees of depth six. We measure the results using the same mean per-class accuracy score as used in [27].

The results for 30 and 1500 training images are shown in Table 2 and one instance is shown in Figure 9. Even without adding pairwise interactions, our learned unary weights already outperform the random forest classifier [27]. When adding more interactions (+1, +1,20, +1,5,20), the performance increases because dense pairwise interactions can represent implicit size preferences for the body parts. Likewise, when adding conditionality (MRF to DTF), the performance improves. The best performing model is our DTF with large structure (+1,5,20) and almost 1.5 million free parameters. It is trained in only 22 minutes and achieves 27.35% mean per-class accuracy. For the same setup of 30 and 1500 training images, the original work [27] reports mean per class accuracies of 14.8% (30 train) and 34.4% (1500 train), but reports an impressive 56.6% with 900k training images, trained for a day on a 1000 core cluster.

An example of a learned pairwise interaction is shown in Figure 10, demonstrating that the improved performance of the DTF can be attributed to the more powerful interactions that are allowed to take the image into account. We report more results in the supplementary materials.

Model	Measure	[27]	unary	+1	+1,20	+1,5,20
MRF 30	avg-acc	14.8	21.36	21.96	23.64	24.05
	runtime	1m	3m18	3m38	10m	10m
	weights	-	176k	178k	183k	187k
DTF 30	avg-acc	-	-	23.71	25.72	**27.35**
	runtime	-	-	5m16	17m	22m
	weights	-	-	438k	951k	1.47M
MRF 1500	avg-acc	34.4	36.15	37.82	38.00	39.30
	runtime	6h34	*	*	*	$(30h)*$
	weights	-	6.3M	6.2M	6.2M	6.3M
DTF 1500	avg-acc	-	-	39.59	40.26	**41.42**
	runtime	-	-	*	*	$(40h)*$
	weights	-	-	6.8M	7.8M	8.8M

Table 2. Body-part recognition results: mean per-class accuracy, training time (4 cores, 8 threads), and number of parameters. (*) We did not obtain reliable runtimes for the 1500 image runs, as multiple jobs have been running in parallel on the machines used.

6. Conclusion

We have introduced Decision Tree Fields as flexible and accurate models for image labeling tasks. This accuracy is achieved by being able to represent complex image-dependent structure between labels. Most importantly, this expressiveness is achieved without the use of latent variables and therefore we can learn the parameters of our model efficiently by minimizing a convex function.

References

[1] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of Markov random fields for segmentation of 3D scan data. In *CVPR*, 2005. 2

[2] D. Batra, R. Sukthankar, and T. Chen. Learning class-specific affinities for image labelling. In *CVPR*, 2008. 2, 6

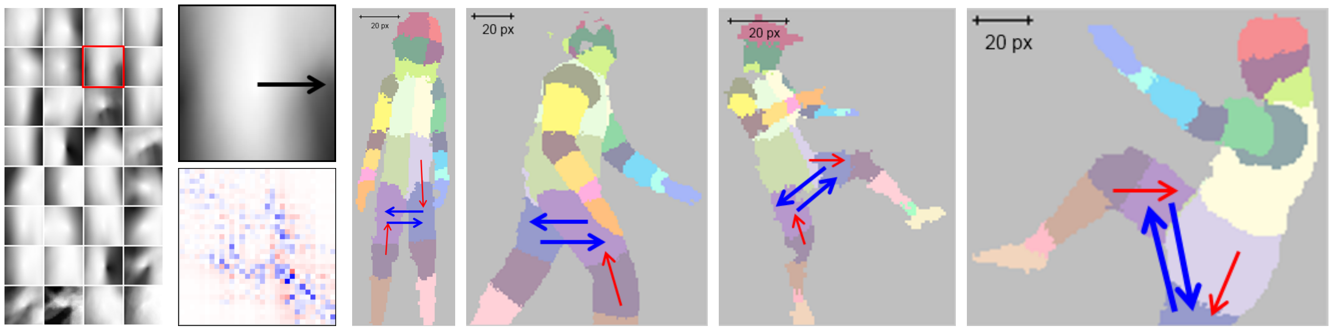

Figure 10. **Illustrating one learned horizontal interaction (20 pixels apart):** The left figure shows the average depth-normalized silhouette reaching one of the 32 leaf nodes in the learned decision tree. We select one leaf (marked red box, enlarged) and show the corresponding effective 32 × 32 weight matrix obtained by summing the learned weights along the path from the root to leaf node. The conditional interaction can be understood by visualizing the most attractive (blue) and most repulsive (red) elements in the matrix. We superimpose arrows for the two most attractive and repulsive interactions on test images (right). The first and second pose exemplify how left and right upper parts of the legs can appear 20 pix to the right of each other in a way that matches the pattern of the leaf. While a configuration like shown in the third and fourth pose is plausible, it does not fit the leaf pattern and thus the interaction is not active.

[3] J. Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrica*, (64):616–618, 1977. 4

[4] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In *ECCV*, 2004. 1, 3

[5] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV*, 2001. 1, 3

[6] L. Breiman. Random forests. *Machine Learning*, 45(1), 2001. 2

[7] T. Chen. *Chinese Calligraphy*. Cambridge University Press, 2011. 7

[8] T. S. Cho, N. Joshi, C. L. Zitnick, S. B. Kang, R. Szeliski, and W. T. Freeman. A content-aware image prior. In *CVPR*, 2010. 2

[9] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *PAMI*, 6:721–741, 1984. 1

[10] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1), 2006. 2

[11] S. Glesner and D. Koller. Constructing flexible dynamic belief networks from first-order probabilistic knowledge bases. In *ECSQARU*, 1995. 2

[12] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009. 2

[13] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *CVPR*, 2004. 1

[14] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. 1, 2, 4

[15] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell*, 28(10):1568–1583, 2006. 1, 5

[16] V. Kolmogorov and Y. Boykov. What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. In *ICCV*, 2005. 1

[17] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001. 1, 3

[18] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, 2009. 2

[19] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer, 1995. 1, 3

[20] S. Nowozin and C. H. Lampert. Global connectivity potentials for random field models. In *CVPR*, 2009. 2

[21] N. Payet and S. Todorovic. $(\mathrm{RF})^2$ – random forest random field. In *NIPS*, 2010. 2

[22] M. Prasad, A. Zisserman, A. W. Fitzgibbon, M. P. Kumar, and P. H. S. Torr. Learning class-specific edges for object detection and segmentation. In *ICVGIP*, 2006. 2

[23] S. Roth and M. J. Black. Steerable random fields. In *ICCV*, 2007. 1

[24] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph*, 23(3):309–314, 2004. 3

[25] P. Schnitzspan, S. Roth, and B. Schiele. Automatic discovery of meaningful object parts with latent CRFs. In *CVPR*, 2010. 2

[26] T. Sharp. Implementing decision trees and forests on a gpu. In *ECCV*, 2008. 2, 5

[27] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*, 2011. 2, 7

[28] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008. 2, 3, 4, 5

[29] 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), 2007. 1, 3, 6

[30] C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In *Introduction to Statistical Relational Learning*, chapter 4. 2007. 1, 4

[31] 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. *PAMI*, 30(7):1068–1080, 2008. 1

[32] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs using graph cuts. In *ECCV*, 2008. 1, 2

[33] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: a large margin approach. In *ICML*, 2005. 2

[34] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *PAMI*, 2010. 3

[35] S. V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In *ICML*, 2006. 1

[36] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2), 2008. 1

[37] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw*, 23(4):550–560, 1997. 4