Stochastic Segmentation Trees for Multiple Ground Truths

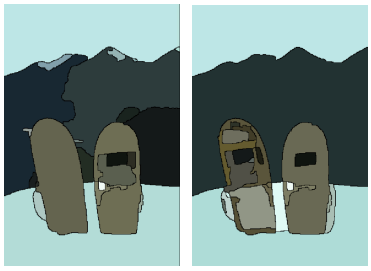
Jake Snell
University of Toronto
Vector Institute

Richard S. Zemel
University of Toronto
Vector Institute
Canadian Institute for Advanced Research

Abstract

A strong machine learning system should aim to produce the full range of valid interpretations rather than a single mode in tasks involving inherent ambiguity. This is particularly true for image segmentation, in which there are many sensible ways to partition an image into regions. We formulate a tree-structured probabilistic model, the *stochastic segmentation tree*, that represents a distribution over segmentations of a given image. We train this model by optimizing a novel objective that quantifies the degree of match between statistics of the model and ground truth segmentations. Our method allows learning of both the parameters in the tree and the structure itself. We demonstrate on two datasets, including the challenging Berkeley Segmentation Dataset, that our model is able to successfully capture the range of ground truths and to produce novel plausible segmentations beyond those found in the data.

(a) Image (b) Ground Truth 1 (c) Ground Truth 2



(d) Sample 1 (e) Sample 2

Figure 1: Multiple ground truths from the Berkeley Segmentation Dataset and samples from our model. Stochastic segmentation trees produce segmentations that generalize through novel combinations of the level of detail in different areas of the image.

1 INTRODUCTION

Humans have a remarkable ability to resolve ambiguity in real-world situations. For example, a skilled interpreter knows that many valid translations of a sentence to a target language exist, and which is the most appropriate depends on context. Yet even in the absence of disambiguating information, humans are able to internally represent a wide range of plausible interpretations when presented with a stimulus. As the central aim of AI is to build machines that can reason about the world in a human-like way, strong AI systems should also be able to maintain a distribution over plausible outputs when confronted with an ambiguous task. Many current machine learning approaches are unable to do this despite the prevalence of ambiguity in domains such as natural language parsing, machine translation, image caption generation, and visual scene understanding.

Tasks involving ambiguity can often naturally be cast as structured output learning problems, in which outputs have internal dependencies. Examples of structured outputs are natural language sentences or paths in a graph. It is often infeasible to independently model each element of the exponentially large structured output spaces typically considered in such problems. Fortunately, graphical models provide a powerful and flexible framework for building rich distributions over structured

output spaces. This is a boon for tasks involving ambiguity as they tend to naturally have multiple ground truths, each reflecting a different yet correct interpretation of the input. For example, the Berkeley Segmentation Dataset (BSD) (Martin et al., 2001; Arbelaez et al., 2011) contains segmentations of natural images collected from multiple human annotators. Due to variability in interpretation or attention, some annotators provided more detailed segmentations or emphasized different contour types than others (Hou et al., 2013). See Figure 1 for an example of such variation.

In this work we introduce the *stochastic segmentation tree* (SST) model, which treats multiple ground truths as samples from a target distribution and learns to predict a distribution specific to the given image. Motivated by the observation that biological visual systems parse scenes according to a hierarchy (Sharon et al., 2006), the model first builds a tree-structured latent region hierarchy of the image. It then predicts a distribution over binary variables corresponding to nodes in the tree that indicate which regions in the tree should be grouped in the same segment.

We make a number of contributions in this paper:

- We introduce an aim that to our knowledge is novel, of producing for a single test input multiple valid structured outputs that directly capture variations in ground truths (GTs).
- We formulate a probabilistic model, a *stochastic segmentation tree*, that represents a distribution over segmentations of a given image.
- We develop a method of forming a tree specific to a given image, and assigning probabilities to its nodes, from a base set of pixels or superpixels (groupings of pixels).
- We define a novel objective that quantifies the degree of match between the model and ground truth segmentations, and formulate an optimization method that allows learning of both the parameters in a tree and the tree structure itself.

2 RELATED WORK

The first line of research related to our SSTs concerns learning systems that produce multiple structured outputs. Many of these approaches can be viewed as structured output generalizations of common multi-label prediction, in which an input can be associated with any number of outputs. One way this can be done is to redefine the output space to be over sets rather than single outputs (Lampert, 2011). Another strategy is to train an ensemble of models to produce multiple hypotheses at test time, either independently (Guzman-Rivera et al., 2012) or in a cascade (Dey et al., 2015). Multiple outputs can also be produced by applying a specialized inference procedure, such as Diverse M-Best MAP (Batra et al., 2012; Yadollahpour et al., 2013; Kirillov et al., 2015) or Diverse Beam Search (Vijayakumar et al., 2016). These approaches have some overlap with our SSTs but differ in aims and formulation. None of these methods produce alternative valid interpretations given an input but instead predict a set of outputs such that one of them is good. Others have attempted to model diverse (and thereby multiple) outputs more directly via Structured DPPs (Kulesza et al., 2012; Gillenwater et al., 2012).

Our model is closely related to approaches that build a graphical model whose structure is conditioned on input data. For example, Wong & Mooney (2006) learn a set of context-free grammar rules which are used to produce a parse tree for a given natural language sentence. They also learn model parameters that define a distribution over meaning representations for the predicted parse tree. Their graphical model parameters are globally learned whereas ours are predicted by the model given the input. The machine translation model of Tromble et al. (2008) forms a distribution over sentences via paths through a translation lattice given a source sentence.

Our model also relates to structured output models, but differ in the training and evaluation criteria. Structured output models are conventionally trained with maximum likelihood given the ground truths for each input. Models are typically evaluated, however, via a task-specific loss comparing a single model output against the ground truths. Examples of task losses include mean average precision for rankings and BLEU (Papineni et al., 2002) for text generation. This disconnect between training and test has spurred many attempts to minimize task loss at training time (Hazan et al., 2010; Song et al., 2015; Shen et al., 2015; Ranzato et al., 2015). These training procedures often improve task loss performance at test but require additional loss-augmented inference or dynamic programming steps during training. In contrast, our method constructs a target distribution from the ground truths and learns to minimize the distance between the model and target distributions. When the model is amenable to such distance computation, this is a more efficient way to learn than task loss minimization since no inference is required. Moreover, the evaluation of model outputs against multiple ground truths via a task loss depends on specific characteristics of the (often few in number) ground truths. BLEU, for example, compares a model generated sentence against n-grams found in ground truth sentences. Yet replacing words in the ground truth sentences with direct synonyms would likely be equally plausible outputs. Using

domain knowledge to construct a target distribution invariant to such low-level variations is a more direct way to train and evaluate models, and is the approach we take in this work.

We now review approaches specific to image segmentation. The goal of image segmentation is to partition an image into meaningful regions. Several task losses are used to measure segmentation performance; one of the most popular is the Probabilistic Rand Index (Unnikrishnan & Hebert, 2005), a generalization of Rand Index (Rand, 1971) to the single-output multiple ground-truth setting. A common approach to image segmentation is to form a hierarchical region grouping, in which regions of finer-scale segmentations are subregions of those at coarser scales, and then cut the tree to obtain an output segmentation. The gPb-OWT-UCM (Arbelaez et al., 2011) algorithm (called UCM, hereafter) utilizes the oriented watershed transform on gPb (Maire et al., 2008) boundaries, followed by greedy merging of regions. It forms a weighted boundary map such that thresholding it at any level produces a valid segmentation, and the threshold value controls the scale. Jain et al. (2011) propose a method that learns to agglomerate superpixels into hierarchies, by learning a similarity function based on ground-truth clustered data. The agglomerative region clustering method (Ren & Shakhnarovich, 2013) uses a learned boundary probability model to merge regions until the estimated probability of merging is below a threshold. Arbelaez et al. (2014) propose a multiscale combinatorial grouping that uses a fast normalized cuts algorithm to produce a hierarchical segmentation that leverages information across different scales.

In general, these methods provide alternative approaches to construct the visual hierarchy from the image; our primary aim is to utilize this tree structure to generate alternative segmentations. The two approaches that most resemble our model are first the work by Ion et al. (2011), which composes multiple figure-ground hypotheses obtained by applying constraints at different locations and scales, into multiple larger interpretations of the entire image. More recently, Hu et al. (2015) propose a probabilistic generative model for segmentations, which can efficiently generate segmentations of varying granularity; unlike our SSTs this work does not propose a novel tree construction process, and more importantly, is trained via maximum likelihood as opposed to matching the set of human annotations.

3 SST MODEL

Our SST model produces multiple segmentation hypotheses from a single image in three steps: (1) a tree structure is constructed; (2) probabilities are associated with nodes in the tree; (3) sample segmentations are generated from the tree based on the node probabilities. Here we detail these steps, initially assuming that the tree structure is given. We first describe how nodes in a tree relate to regions of an image, and how a binary labeling of nodes in such a tree corresponds to a segmentation of the image. We then present the probabilistic model over segmentations based on these binary labelings, and a metric comparing the model samples to ground-truth segmentations of an image that can be used to evaluate a multiple segmentation system. This metric is then the objective utilized in learning the network that predicts the node probabilities from an image at test time. Finally at the end of this section we describe our own procedure for generating trees, which is also based on this metric.

3.1 PROBABILISTIC TREE MODEL

A segmentation can be represented as a vector of numbers, where each pixel is assigned a particular region index $r \in \mathcal{R}$. In such a representation, pixels assigned the same index r are considered to be grouped together. An alternative representation of the same information is as a symmetric binary matrix $\mathbf{S}$, where $S_{ij} = 1$ if pixel i and j are assigned to the same region, and $S_{ij} = 0$ otherwise. We adopt this representation here.

We also utilize a third representation of a segmentation, with respect to a *segmentation tree*. A segmentation tree has M leaves, one for each of M non-overlapping and complete image superpixels. A superpixel $m \in \{1, ..., M\}$ represents a grouping of contiguous pixels in the image; we denote the set of pixels assigned to superpixel m as $C(m)$. As a slight abuse of notation, we use m to denote both a superpixel and its corresponding leaf in the tree. Each internal node k represents a grouping of pixels that is the union of pixels assigned to its children: $C(k) = C(\ell(k)) \cup C(r(k))$, where $\ell(k)$ and $r(k)$ are the left and right child of k, respectively. This assignment of nodes to left and right children of parent nodes define the tree structure.

Given a particular tree structure, a segmentation is a set of binary values $\mathbf{t}$, where each leaf m has $t_m = 1$, and the binary value t_k associated with each internal tree node k represents whether or not the set of pixels in its left child and the pixels of the right child belong to the same image region:

$$t_k = 1 \Rightarrow S_{ij} = 1, \ \ \forall i \in C(\ell(k)), j \in C(r(k)) \quad (1)$$

These binary values are constrained to ensure that the values in the pairwise pixel matrix $\mathbf{S}$ correspond to a valid segmentation. This is achieved by forcing $t_k = 0$ if either $t_{\ell(k)} = 0$ or $t_{r(k)} = 0$. This constraint results in tree labelings in which any path that starts at a leaf and

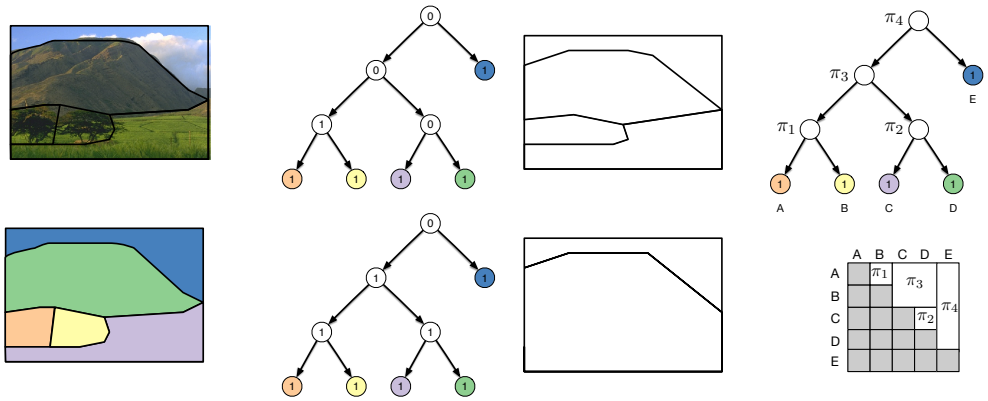


Figure 2: Left: image and color-coded superpixels. Middle: two sample segmentations, with corresponding binary values ($\mathbf{t}$) of the segmentation tree. Right: each node k in the tree has an associated marginal probability π_k, which governs a specific region of the pairwise segmentation matrix.

ends at the root has an initial value of 1, and if the value switches to 0 along the path then it stays at 0 through the root.

The last node along this path with a value of 1 defines a sub-tree of superpixels that all belong to the same image region, which corresponds to a block in the segmentation matrix. The relationship between the tree labeling $\mathbf{t}$ and the matrix $\mathbf{S}$ is illustrated in Figure 2.

In order to represent the variety of segmentations of a given image, with different levels of detail in different regions, we formulate a probabilistic model based on a leaf-to-root sampling scheme. The probabilities are assigned to nodes in the tree, which directly correspond to probabilities in the segmentation matrix. These equations define the SST probabilistic model:

$$P(t_k = 1 | \mathbf{t}_{-k}) \quad = \quad \begin{cases} p_k & \text{if } t_{\ell(k)} = t_{r(k)} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{t}_{-k}$ is the vector of binary values associated with the nodes below k in the tree. Given this definition, and a sampling scheme that samples binary values t_k starting at the leaves and moving up the tree, then it can be shown that the marginal probability π_k that node k takes on value 1 is a product of the local probability of that node and the marginals of its children:

$$\pi_k \equiv P(t_k = 1) = \prod_{d \in subTree(k)} p_d \;=\; p_k \pi_{\ell(k)} \pi_{r(k)} \tag{3}$$

where $subTree(k)$ are all the nodes in the sub-tree rooted at node k.

The probabilistic model plays an important role in our system, specifying a distribution over segmentations of an image, which can be sampled to generate hypothesized segmentations. An example probability vector $P(\mathbf{t})$ for a given tree, and some corresponding samples are illustrated in Figure 2.

3.2 EVALUATING AN SST

The aim of the stochastic segmentation tree is to produce segmentations such that their statistics match the statistics of the reference segmentations for a given image. One metric that is well-suited to this goal is the *Hellinger distance*, which is a metric for comparing two distributions.

In our setting we utilize the marginal pairwise probabilities as the underlying probabilities to be compared via the Hellinger distance. For the model, the marginal probability that pixel i is in the same segment as pixel j is captured by π_k, where $i \in C(\ell(k))$ and $j \in C(r(k))$, or vice versa. Given a set of ground-truth segmentations $\{\mathbf{S}^1, ..., \mathbf{S}^U\}$, the target marginals q_{ij} can be computed easily: $q_{ij} = \frac{1}{U} \sum_u S^u_{ij}$. The squared-Hellinger distance $D(\pi, q)$ is defined to be:

$$1 - \frac{1}{\binom{N}{2}} \left(D_0 + \sum_{\substack{k, i \in C(\ell(k)) \\ j \in C(r(k))}} \left[\sqrt{\pi_k q_{ij}} + \sqrt{(1-\pi_k)(1-q_{ij})} \right] \right) \tag{4}$$

where N is the number of pixels. Here D_0 is the summed Hellinger distance of each of the super-pixels, which sums over the SST leaves of any errors in the super-pixellation; these errors are constant and cannot be repaired by the SST.

We propose that matching pairwise statistics is an appropriate goal for the multiple-segmentation scenario. Intuitively, the metric considers all pairs of pixels in the image, and compares how often that pair are grouped together by humans versus the model. An alternative is

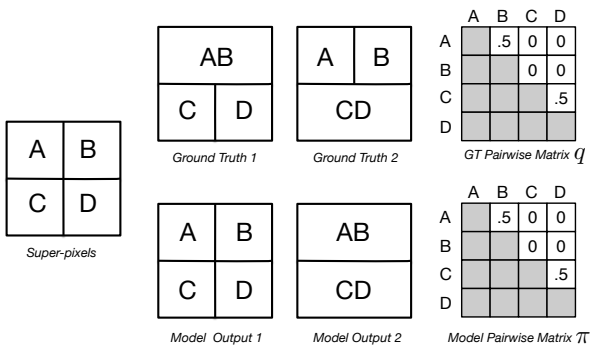


Figure 3: Illustration of the properties of the Hellinger distance as it relates to segmentations. In this scenario, there are four superpixels and two ground truth segmentations. When viewed in the pairwise matrix form, both the ground truth and model distributions are identical despite the fact that the model outputs segmentations not found in the ground truths. We hypothesize that these are sensible generalizations.

to attempt to match higher order statistics between the model and ground-truth segmentations, e.g., triplets of pixels, but given that typically only a small reference set of segmentations are available, these higher-order statistics are not as reliable. A visualization of the properties of the Hellinger distance is shown in Figure 3.

3.3 NETWORK TO ESTIMATE NODE MARGINALS

If the reference segmentations are available, the node probabilities and their corresponding marginals for a given tree structure can be optimized (e.g., via gradient descent) to minimize the Hellinger objective (Equation 4). At test time, however, with no reference segmentation these node marginals must be estimated from the image. We formulate a neural network that estimates node marginals, and optimize it to minimize the Hellinger objective on the training set. Then given a tree for a test image, we can estimate marginals, and sample segmentations.

Our network to estimate the marginals π_k for any node k is a three layer feed-forward neural net. The inputs to the network consist of visual features extracted from regions corresponding to $\ell(k)$ and $r(k)$, as well as a small number of geometric features of the regions, including centroid location and size. The first layer of the network applies the same weights to these features of the left and right regions. The second layer takes as input the concatenation of the hidden left and right representations with their element-wise squared difference. The third and final layer connects to the network output, a single sigmoid unit, which is the estimated marginal probability for the $\ell(k)$, $r(k)$ merge. The same network is used to predict merge marginal probabilities throughout the entire tree.

Learning is performed via SGD with backpropagation, to minimize the Hellinger distance with respect to the parameters of our model. This is made simpler by the fact that $D(\pi, q)$ decomposes over nodes of the tree. The learned network predicts marginal probabilities independently for each node and therefore it is possible that the network's predictions are not consistent with our tree assumptions from Section 3.1. For example, if the marginal probability predicted at a certain node is greater than that predicted at either of its children, there will be no setting of the conditional node probabilities $\mathbf{p}$ consistent with the network outputs that also satisfies Equation 3. Thus as a final step, we perform a separate constrained optimization using L-BFGS (Liu & Nocedal, 1989) to find the setting of $\mathbf{p}$ that has minimal Hellinger distance to the marginals outputted by the network while still obeying our tree assumptions. We then recompute the marginal probabilities corresponding to this setting of $\mathbf{p}$ using Equation 3 and treat these reconciled marginals as the final outputs of the model.

Sample segmentations can be generated from the model using a simple bottom-up procedure. The non-terminal nodes in the tree are traversed in bottom-up order. For each node k, draw $x_k \sim \text{Uniform}(0, 1)$. If $x_k < p_k$ and $t_{\ell(k)} = t_{r(k)} = 1$, then $t_k \leftarrow 1$. Otherwise, $t_k \leftarrow 0$. This setting of $\mathbf{t}$ defines a segmentation that is a sample from the model (see Figure 2 for a example $\mathbf{t}$ and the corresponding segmentation).

3.4 TREE CONSTRUCTION

The discussion above assumed the tree structure was given, as could be provided by some existing segmentation tree algorithm such as UCM (Arbelaez et al., 2011) or ISCRA (Ren & Shakhnarovich, 2013). Here we extend our formulation to also construct a tree for a given image. Our tree construction algorithm, called SST-SWAPTREE, begins with an arbitrary tree structure and seeks to iteratively improve it by making moves within the space of tree structures. We consider swaps, in which the subtree rooted at a node in the tree is exchanged with the subtree rooted at another node. Refer to Figure 4 for an illustration of a sample swap. Alternate strategies for constructing trees, such as agglomerative clustering, are possible but we opt to iteratively improve tree structures rather than building them from scratch for two main reasons. First, maintaining valid tree structures makes it easier to track performance throughout the tree building process. Second, it is simpler to estimate the effect of a move on the entire tree structure because the global con-

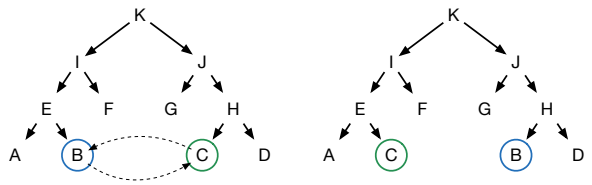


Figure 4: An example swap. **Left**: The tree structure before swapping B and C. **Right**: The structure after the swap. Note that when non-leaf nodes are swapped, their descendants also accompany them in the swap.

text is known, as opposed to agglomerative clustering in which part of the tree structure does not yet exist.

At each iteration of SST-SWAPTREE, the first step involves enumerating allowable swaps. A swap between nodes is allowable if neither is a descendant of the other, as otherwise the resulting structure is ill-defined. It also must preserve the property that children of any non-terminal node in the tree are neighboring. Both of these properties can be checked efficiently (see the supplementary material for details).

The next step involves scoring a swap. The key idea in this step is that any tree can be evaluated based on the minimum achievable Hellinger distance between the tree and the ground-truth segmentations. Hence a swap can be scored based on how it much it lowers this minimum distance. At training time we have access to the ground truth segmentations, and so the minimum achievable Hellinger distance for a given structure can be easily computed via numerical minimization of $D(\pi, q)$ with respect to π. At test time however this is not possible, so instead we utilize our learned network that estimates node marginals to approximate the change in achievable Hellinger distance effected by a swap, which determines its score. Every iteration of the algorithm greedily selects the highest scoring swap and produces a new tree structure. This procedure is repeated for a fixed number of iterations chosen based on validation performance. Space does not permit a detailed description here of our SST-SWAPTREE algorithm for enumerating and scoring swaps, and so further details can be found in the supplementary material.

4 EXPERIMENTS

In order to assess the effectiveness of SSTs in producing multiple plausible segmentations, we learned models on two segmentation datasets: the Berkeley Segmentation Dataset 500 (BSDS500) (Arbelaez et al., 2011) and the Penn-Fudan pedestrian parsing dataset (Bo & Fowlkes, 2011; Wang et al., 2007). In this section, we first describe the datasets we used. We then provide details of our model training procedure and show experimental results.

4.1 DATASETS

The BSDS500 dataset is a natural fit for our task because it contains multiple human-annotated ground truth segmentations for each image. The Penn-Fudan pedestrian dataset is a semantic image dataset that we adapt for our purposes by generating four synthetic GT segmentations per example, each of which represents a different prototypical parse of the image. We generate these GTs based on merging different body parts in different segmentations. Details of the merging algorithm, along with visualizations of several ground truths and corresponding images are shown in the supplementary material.

The BSDS500 dataset contains predefined training, validation, and test splits of 200, 100, and 200 examples, respectively. The Penn-Fudan dataset contains only 169 examples, so we randomly created five splits with 20% of the examples as test and the remainder divided into training and validation in an 80%-20% ratio. The reported evaluation results for this dataset are averaged over test predictions across the five splits.

4.2 SST TRAINING DETAILS

We took the regions output by the UCM algorithm, prior to thresholding, as the leaves of our trees. For BSDS500 this yielded a large number of leaves per image, so we thresholded the weighted contour map for each example independently such that 100 regions remained. All methods, including baselines, used the same base-level regions.

Our network for estimating marginal probabilities used two sets of features as input. The first are visual features computed by average-pooling activations from the fourth convolutional layer of the VGG-19 network (Simonyan & Zisserman, 2014). The second set were geometric features regarding the regions. Both sets of features have the benefit that the features of a parent region can efficiently be computed given features of the child regions.

The marginal probability estimation network was trained with SGD using the Adam optimizer (Kingma & Ba, 2014). Early stopping was done on the Hellinger distance of the model merge probability assigned trees on the validation set. Dropout was used on the first two layers of the network for regularization. Further training and architectural details are contained in the supplementary material.

4.3 RESULTS

We evaluate the learned SST models along with several baselines, which we describe here. We distinguish between methods that produce a single output and those that generate multiple outputs.

Of the single output baselines, the simplest is SUPERPIXELS, the segmentation consisting of the starting base-level regions themselves. BOYKOV-JOLLY is a baseline inspired by Boykov & Jolly (2001) in which strengths between regions are computed as a monotonically increasing function of the squared difference in mean intensity between them. This induces a tree structure, which can be thresholded at any strength in order to form a segmentation. UCM SINGLE is the standard single segmentation produced by UCM, based on thresholding the hierarchy at the threshold yielding the best performance on the validation set.

For multiple-output baselines, we consider an extension of UCM SINGLE, in which a Gaussian distribution over thresholds is constructed. Multiple segmentations can be generated by first sampling a threshold from this Gaussian, and then generating a segmentation by merging regions until this threshold is reached. The mean of this Gaussian is set to the corresponding UCM SINGLE threshold and the variance is tuned on the validation set. We call this extension UCM MULTIPLE.

We compare these baselines to two methods utilizing SSTs. SST-UTREE is our model trained using the UCM tree structures as described in Section 4.2. SST-SWAPTREE is a modified version of the procedure described in Section 3.4 to iteratively improve the trees generated by UCM. The only modification we made was to use the true node marginals when choosing swaps to make. As such these tree structures are oracle trees in that they can only be computed if the optimal π_k^* is known at test time. This tests whether our tree construction procedure produces better trees, provided the $\hat{\pi}$ estimator is sufficiently accurate. The final Hellinger distance computed on the tree structures does however use the network marginal outputs to assign probabilities to each node in the procedure described in Section 3.3.

We also evaluate the methods on a multiple output variant of the Rand Index that we call *matching Rand Index (MRI)*. For each method a set of candidate segmentations is first generated by sampling from the model. Then an optimal matching is found between these and the ground-truth segmentations; this can easily be computed via a max-weighted bipartite matching algorithm. Each ground truth matches the segmentation from the candidate set with the greatest Rand Index with respect to itself, subject to the constraint that no two ground truths match the same model output. The MRI is then the mean Rand Index of the matched segmentations. The MRI results are shown in Table 2. For methods that output a single segmentation we report its average Rand Index to the ground-truths; MRI simplifies to probabilistic Rand Index (PRI) (Unnikrishnan & Hebert, 2005) in this setting. For the multiple output methods we include results with both 50 and 100 samples drawn from the respective models. SST significantly outperforms the baselines here even though it was trained to optimize Hellinger distance, demonstrating that Hellinger is a sensible objective for training models in a multiple output setting. Also, the SST-SWAPTREE MRI shows the gains obtained by improving the tree structure.

Table 1: Test Hellinger distances (lower is better). The first methods all output a single segmentation, while the others output several.

Method	BSDS 500	Penn-Fudan
SUPERPIXELS	0.1451	0.3485
BOYKOV-JOLLY	0.1447	0.3172
UCM SINGLE	0.1315	0.2974
UCM MULTIPLE	0.1315	0.2971
SST-UTREE	0.0864	0.2070
SST-SWAPTREE	**0.0689**	**0.1901**

Table 2: Test matching Rand Index (higher is better).

Method	Samples	BSDS 500	Penn-Fudan
SUPERPIXELS	1	0.8107	0.6366
BOYKOV-JOLLY	1	0.8111	0.6612
UCM SINGLE	1	0.8247	0.6813
UCM MULTIPLE	50	0.8687	0.7037
SST-UTREE	50	0.8786	0.7160
SST-SWAPTREE	50	**0.9116**	**0.7726**
UCM MULTIPLE	100	0.8718	0.7072
SST-UTREE	100	0.8813	0.7180
SST-SWAPTREE	100	**0.9148**	**0.7792**

Segmentations sampled from the SST-UTREE model are visualized in Figure 5 for BSDS500. We observe that the samples vary across regions of the image, capturing the idea of generating segmentations at different levels of granularity.

4.4 APPLICATION OF SST: SEMANTIC SEGMENTATION

The stochastic segmentation tree provides an informative representation of the image structure. The tree, with each node assigned its own well-calibrated probability of the associated merge, describes a rich distribution over

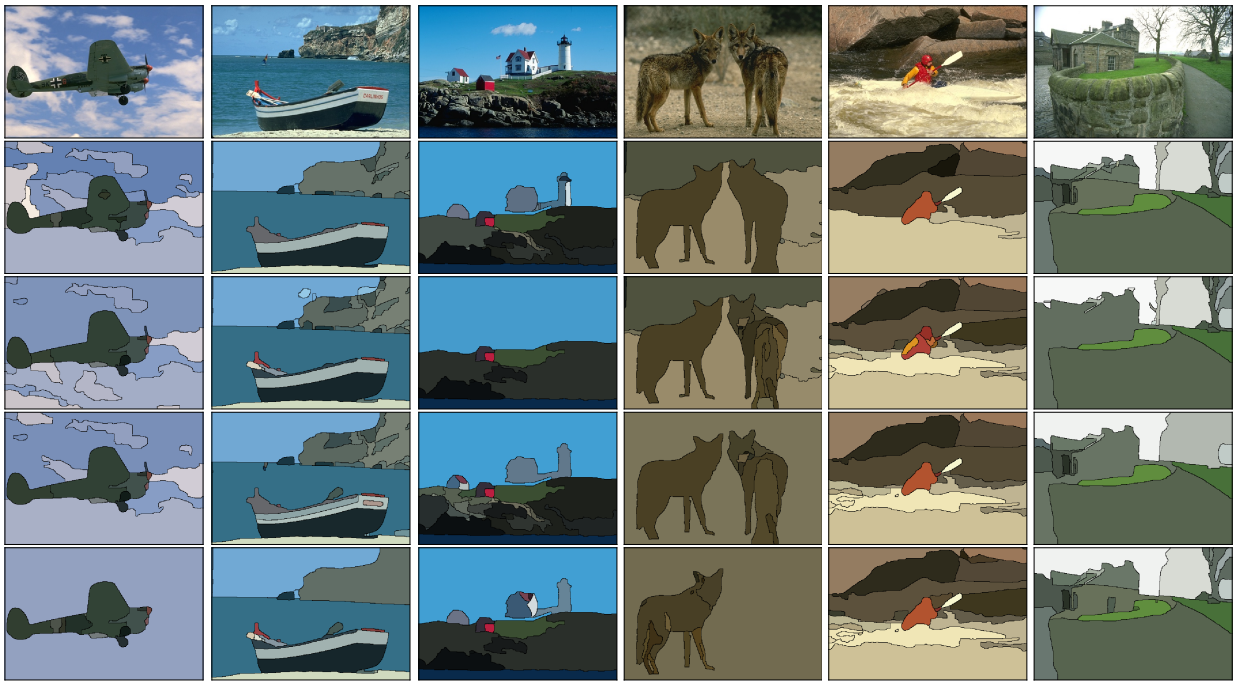


Figure 5: Samples from the SST-UTREE model on BSDS500. From top to bottom: image, samples 1-4 from the model. All images are from the test set.

segmentations. In addition, there are a number of potential applications of SSTs. Any algorithm $\mathcal{A}$ that uses superpixels as inputs could benefit from this rich representation. If $\mathcal{A}$ wants k superpixels, the SST can produce multiple instances with approximately the same number, but with varying granularity in different regions of the image. Other existing tree-based methods do not have the same ability. For example, UCM generates alternative segmentations, but these obey a fixed ordering, so they cannot produce higher granularity in one region and lower in another in one segmentation, and vice versa in a second segmentation. Running $\mathcal{A}$ on each of these superpixel inputs provides a natural measure of uncertainty in the algorithm's output.

Here we explore one such algorithm, semantic segmentation, that utilizes superpixels as inputs. In semantic segmentation, the goal is to assign each pixel in an image one of a fixed set of labels. We consider a scenario in which an SST is used to generate multiple segmentations, which are used as superpixel input to a semantic segmentation algorithm. In order to decouple the effects of the specific segmentation algorithm from the performance of the SST as a superpixel generator, we consider an oracle setting in which each region is assigned a single label that gives the highest segmentation accuracy as measured by Hamming distance or intersection-over-union (IOU) between the semantic segmentation and the ground truth.

Our experiments use the Penn-Fudan dataset, as they already contain semantic segmentation labels. We varied the sigmoid output bias of our SST-UTREE model in order to encourage it to generate superpixelations with various numbers of regions per image. We chose the biases to produce roughly 30, 50, 80, and 100 superpixels per image, and computed both the oracle segmentation accuracy and number of segments of 1,000 samples from our model on each test example. The results are shown in Figure 6 for Hamming distance and in Figure 7 for mean intersection-over-union (IOU). IOU was computed for each of the 12 non-background classes in the Penn-Fudan dataset and then averaged across classes to produce mean IOU. For comparison, we also display the oracle segmentation accuracy of the UCM SINGLE algorithm over a range of thresholds and the BOYKOV-JOLLY baseline. The oracle accuracy of SST-UTREE is higher for each of these scenarios than either UCM SINGLE or BOYKOV-JOLLY. In order to illustrate how the sampled segmentations vary, Figure 8 shows sampled oracle segmentations from the SST-UTREE model that produces roughly 50 superpixels per image.

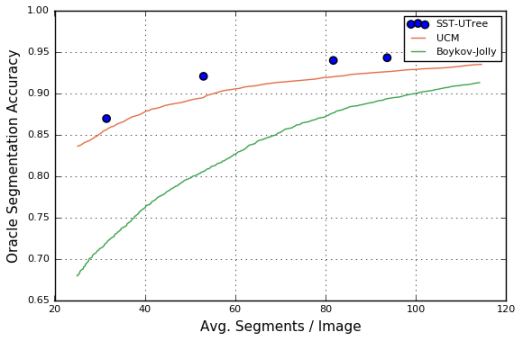


Figure 6: Oracle segmentation accuracy (Hamming) for SST-UTREE, UCM-SINGLE, and BOYKOV-JOLLY on the Penn-Fudan dataset.

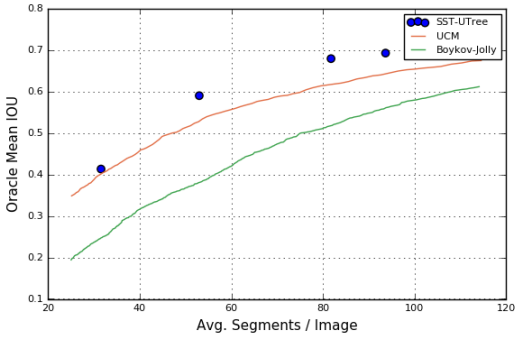


Figure 7: Oracle segmentation accuracy (mean IOU) for SST-UTREE, UCM-SINGLE, and BOYKOV-JOLLY on the Penn-Fudan dataset.

5 CONCLUSION & FUTURE WORK

We have presented a model for producing multiple segmentations. The model extends current hierarchical segmentation approaches by formulating a probabilistic model using its structure. The probabilistic objective, based on the Hellinger distance, provides a fairly simple metric for evaluating and training this model. We formulate a learning approach to estimate merge probabilities for the internal nodes of a given tree. We then utilize this same estimator in order to optimize the tree structure, based on a sequence of sub-tree swaps, which effectively improves our loss function. We tested the model against two strong baselines, the standard Boykov-Jolly as well as a top current segmentation approach. The model provides a significant win on two datasets, including the challenging BSDS500. We also showed its utility in semantic segmentation, as a method of forming superpixels at different levels of granularity.

We are currently investigating a number of directions, including further improvements in the tree construction

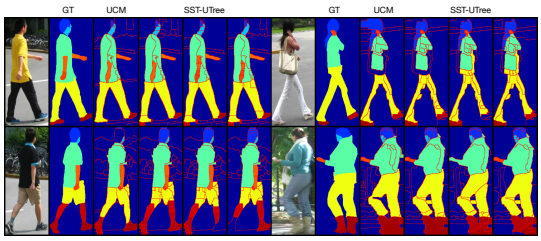


Figure 8: Oracle segmentation visualizations. From left to right: image, ground truth segmentation, UCM-SINGLE oracle segmentation, and oracle segmentations from three samples from SST-UTREE. Superpixels were generated from the $\sim$50 superpixels per image setting for both UCM-SINGLE and SST-UTREE. Superpixels are outlined in red.

procedure. If the complexity of using multiple tree structures can be managed, this should provide a performance boost as well. Multiple trees would allow the model to handle diversity in the segmentations that go beyond that captured in a single tree, such as alternative groupings of ambiguous regions. In addition, allowing the merge probability estimator to take into account more information about regions, such as the level of the tree, and features at multiple levels, may also be beneficial.

The proper treatment of ambiguity is important to the future progress in AI. We believe that predicting the structure and parameters of a graphical model given an input, as we have done here for image segmentation, is applicable to other domains as well. Such models facilitate the minimization of distance to a target distribution constructed from multiple ground truths, which when appropriately built according to the task is a sensible objective. We hypothesize that this approach can be used to improve both training and evaluation of models in a wide range of important domains, particularly in tasks involving ambiguity in computer vision and natural language processing.

Acknowledgements

We thank Chris Williams and Raquel Urtasun for their helpful comments. This research was supported by Samsung and the Natural Sciences and Engineering Research Council of Canada.

References

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. Contour Detection and Hierarchical Image Segmentation. *IEEE PAMI*, 2011.

Arbelaez, P., Pong-Tuset, J., Barron, J., Marques, F., and Malik, J. Multiscale Combinatorial Grouping. In *CVPR*, 2014.

Batra, Dhruv, Yadollahpour, Payman, Guzman-Rivera, Abner, and Shakhnarovich, Gregory. Diverse M-Best Solutions in Markov Random Fields. In *ECCV*, 2012.

Bo, Y. and Fowlkes, C. Shape-based Pedestrian Parsing. In *CVPR*, 2011.

Boykov, Y. and Jolly, M.-P. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *ICCV*, 2001.

Dey, Debadeepta, Ramakrishna, Varun, Hebert, Martial, and Bagnell, J Andrew. Predicting Multiple Structured Visual Interpretations. In *ICCV*, 2015.

Gillenwater, Jennifer, Kulesza, Alex, and Taskar, Ben. Discovering Diverse and Salient Threads in Document Collections. In *EMNLP*, 2012.

Guzman-Rivera, A., Batra, D., and Kohli, P. Multiple Choice Learning: Learning to Produce Multiple Structured Outputs. In *NIPS*, 2012.

Hazan, Tamir, Keshet, Joseph, and McAllester, David A. Direct Loss Minimization for Structured Prediction. In *NIPS*, 2010.

Hou, X., Yuille, A., and Koch, C. Boundary Detection Benchmarking: Beyond F-Measures. In *CVPR*, 2013.

Hu, Shell X, Williams, Christopher KI, and Todorovic, Sinisa. Tree-Cut for Probabilistic Image Segmentation. *arXiv preprint arXiv:1506.03852*, 2015.

Ion, A, Carreira, J, and Sminchisescu, C. Image Segmentation by Figure-Ground Composition into Maximal Cliques. *ICCV*, 2011.

Jain, V., Turaga, S., Briggman, K., Helmstaedter, M., Denk, W., and Seung, H. S. Learning to Agglomerate Superpixel Hierarchies. In *NIPS*, 2011.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirillov, Alexander, Savchynskyy, Bogdan, Schlesinger, Dmitrij, Vetrov, Dmitry, and Rother, Carsten. Inferring M-Best Diverse Labelings in a Single One. In *ICCV*, 2015.

Kulesza, Alex, Taskar, Ben, et al. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3):123–286, 2012.

Lampert, C. Maximum Margin Multi-Label Structured Prediction. In *NIPS*, 2011.

Liu, Dong C and Nocedal, Jorge. On the Limited Memory BFGS Method for Large Scale Optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Maire, Michael, Arbeláez, Pablo, Fowlkes, Charless, and Malik, Jitendra. Using Contours to Detect and Localize Junctions in Natural Images. In *CVPR*, 2008.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *ICCV*, 2001.

Papineni, Kishore, Roukos, Salim, Ward, Todd, and Zhu, Wei-Jing. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*, 2002.

Rand, W. Objective Criteria for the Evaluation of Clustering Methods. *JASA*, 12 1971.

Ranzato, Marc'Aurelio, Chopra, Sumit, Auli, Michael, and Zaremba, Wojciech. Sequence Level Training with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06732*, 2015.

Ren, Z. and Shakhnarovich, G. Image Segmentation by Cascaded Region Agglomeration. In *CVPR*, 2013.

Sharon, E., Galun, M., Sharon, D., Basri, R., and Brandt, A. Hierarchy and Adaptivity in Segmenting Visual Scenes. *Nature*, 442(7104):810–813, August 2006.

Shen, Shiqi, Cheng, Yong, He, Zhongjun, He, Wei, Wu, Hua, Sun, Maosong, and Liu, Yang. Minimum Risk Training for Neural Machine Translation. *arXiv preprint arXiv:1512.02433*, 2015.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Song, Yang, Schwing, Alexander G, Zemel, Richard S, and Urtasun, Raquel. Direct Loss Minimization for Training Deep Neural Nets. *arXiv preprint arXiv:1511.06411*, 2015.

Tromble, Roy W, Kumar, Shankar, Och, Franz, and Macherey, Wolfgang. Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *EMNLP*, 2008.

Unnikrishnan, Ranjith and Hebert, Martial. Measures of Similarity. In *IEEE ACV*, 2005.

Vijayakumar, Ashwin K, Cogswell, Michael, Selvaraju, Ramprasath R, Sun, Qing, Lee, Stefan, Crandall, David, and Batra, Dhruv. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

Wang, L., Shi, J., Song, G., and Shen, I. Object Detection Combining Recognition and Segmentation. In *ACCV 2007*, 2007.

Wong, Yuk Wah and Mooney, Raymond J. Learning for Semantic Parsing with Statistical Machine Translation. In *ACL*, 2006.

Yadollahpour, Payman, Batra, Dhruv, and Shakhnarovich, Gregory. Discriminative Re-ranking of Diverse Segmentations. In *CVPR*, 2013.

A DETAILS OF SST-SWAPTREE ALGORITHM

Our tree construction algorithm begins with an arbitrary tree structure and seeks to iteratively improve it by making moves within the space of tree structures. We consider swaps, in which the subtree rooted at a node in the tree is exchanged with the subtree rooted at another node. Refer to Figure 4 for an illustration of a sample swap.

We now present the details of how we efficiently compute and score allowable swaps in order to decide which move to make. This procedure continues for a fixed number of iterations chosen based on validation performance.

A.1 COMPUTATION OF ALLOWABLE SWAPS

We consider only swaps that satisfy two criteria. The first is that a swap must not be between a node and an ancestor of that node. We do not consider such swaps because the subtree rooted at each of two nodes overlap and therefore the position of any shared descendants after the swap is ill-defined. For example, a swap between node B and node I in Figure 4 would not be allowable.

The second criterion is that the swap must preserve the property that the children of any nonterminal node in the tree are neighboring. We call this property the *neighboring-region* property. Informally, the neighboring-region property ensures that for two children of a parent node k to be siblings, at least one of the superpixels corresponding to the left sibling must be adjacent to one of the superpixels corresponding to the right. If this condition is satisfied, we say that Neigh$(C(\ell(k)), C(r(k)))$. More precisely, the neighboring-region property is satisfied by a tree if for every non-terminal node k in the tree, $N(C(\ell(k))) \cap C(r(k)) \neq \varnothing$, where $N(S) = \left(\bigcup_{s \in S} \text{Adj}(s)\right) \setminus S$ denotes the set of superpixels directly adjacent to the superpixels contained in set S but not including the elements of S itself, and Adj(s) denotes the set of superpixels directly adjacent to superpixel s. Note that the neighboring-region property is symmetric in that it could equally well be expressed as $N(C(r(k))) \cap C(\ell(k)) \neq \varnothing$ for each non-terminal node k.

Finding swaps that satisfy the first criterion is simple. We begin by iterating over each node i in the tree. For each node i, we can search for potential swap partners j by traversing the path from i to the root. For each node k along this path (excluding i itself), we search the the subtree rooted at the child of k not belonging to that path for potential swap partners j. For example, if B were being considered for a swap in Figure 4, the path to the root (excluding B) would be E $\rightarrow$ I $\rightarrow$ K. We would therefore search the subtrees rooted at A, F, and J for nodes that could be swapped with B.

Satisfying the second criterion relies on two observations that hold for all swaps between nodes i and j: (a) any node not on the path from i to j will continue to satisfy the neighboring-region property after the swap, and (b) the lowest common ancestor $k = LCA(i, j)$ will maintain the neighboring-region property after the swap. Observation (a) is due to the fact that the regions corresponding to these nodes do not change as a result of the swap. For example, in Figure 4, the path from node B to C is B $\rightarrow$ E $\rightarrow$ I $\rightarrow$ K $\rightarrow$ J $\rightarrow$ H $\rightarrow$ C. A swap between B and C will not affect the subtrees rooted at A, F, G, or D, and therefore we do not need to check the neighboring-region property for these nodes. In addition, if there were any nodes above K in the example tree, they would not need to be checked either since the region corresponding to K in the tree does not change as a result of the swap. Furthermore, the subtrees rooted at i and j themselves do not change as a result of the swap and thus do not need to be checked.

To show observation (b) holds, assume without loss of generality that i is a descendant of $\ell(k)$ and j is a descendant of $r(k)$ (the right child of k). We know that prior to the swap, Neigh$(C(\ell(k)), C(r(k)))$ is true. Thus Neigh$((C(\ell(k)) \setminus C(i)) \cup C(i), (C(r(k)) \setminus C(j)) \cup C(j))$ is also true. For k to satisfy the neighboring-region property after the swap, it must be the case that Neigh$((C(\ell(k)) \setminus C(i)) \cup C(j), (C(r(k)) \setminus C(j)) \cup C(i))$. If $\ell(k) \neq i$, then we know that Neigh$(C(\ell(k)) \setminus C(i), C(i))$, otherwise the subtree rooted at $\ell(k)$ would not have satisfied the neighboring-region property prior to the swap. Similarly, if $r(k) \neq j$, then we know that Neigh$(C(r(k)) \setminus C(j), C(j))$. If both $\ell(k) = i$ and $r(k) = j$, then the neighboring-region property is satisfied trivially after the swap, because otherwise Neigh$(C(\ell(k)), C(r(k)))$ would not be true prior to the swap.

Therefore we must only check the nodes along the path from i to j that are not either the $LCA(i, j)$ or i or j themselves. Let m be such a node along the path from i to $LCA(i, j)$. Without loss of generality, assume that i is a descendant of $\ell(m)$. Define the set of *critical superpixels* in i relative to a node h to be Crit$(i, h) = C(i) \cap N(C(h))$. For m to have satisfied the neighboring-region property prior to the swap, it must be the case that Crit$(\ell(m), r(m)) \neq \varnothing$. If Crit$(\ell(m), r(m)) =$ Crit$(i, r(m))$, i.e. all critical superpixels of $\ell(m)$ relative to $r(m)$ are also contained in $C(i)$, then we say i is *critical* relative to $r(m)$.

If i is critical relative to $r(m)$, we know m will not satisfy the neighboring-region property after a swap involving i unless the region of the replacement node j is neighboring to the region of $r(m)$. We can thus add a *critical constraint* on j due to m that $C(j) \cap N(r(m)) \neq \varnothing$. Any such constraints can be accumulated as we traverse the path from i to $LCA(i,j)$, and we must check that the candidate node j satisfies all constraints. Similarly we must check that i satisfies all critical constraints collected along the path from j to $LCA(i,j)$. If all such constraints are satisfied then the swap between i and j is allowable.

A.2 SWAP SCORING

In this section we outline the algorithm used to assign scores to potential swaps. We first examine the minimum achievable Hellinger distance for a given tree structure. Then we show how to compute the change in minimum achievable Hellinger distance resulting from a swap. Finally, we derive an approximation to the change in Hellinger distance that can be computed given only information known at test time.

A.2.1 Minimum Achievable Hellinger Distance

When iteratively making swaps, we seek to decrease Hellinger distance achievable by the tree. In other words, we wish to decrease

$$\begin{aligned}\min_{\mathbf{p}} D(\pi, q) = 1 - &\frac{1}{\binom{N}{2}} \sum_k \sum_{\substack{i \in C(\ell(k)) \\ j \in C(r(k))}} \left[\sqrt{\pi_k q_{ij}} + \sqrt{(1-\pi_k)(1-q_{ij})} \right] \\ &- \frac{1}{\binom{N}{2}} \sum_m \sum_{\substack{i,j \in C(m) \\ i<j}} \sqrt{q_{ij}}, \end{aligned} \tag{5}$$

where the summation involving k is over nonterminal nodes in the tree, the summation involving m is over leafs of the tree, $\mathbf{p}$ is a vector of probabilities (one for each nonterminal node), and $\pi_k = p_k \pi_{\ell(k)} \pi_{r(k)}$ is the marginal probability of node k.

When scoring swaps it is more convenient to consider minimization directly over $\boldsymbol{\pi}$ because the summations decompose over nodes:

$$1 - \frac{1}{\binom{N}{2}} \sum_m \sum_{\substack{i,j \in C(m) \\ i<j}} \sqrt{q_{ij}} - \sum_k \max_{\pi_k} \left[\sqrt{\pi_k} \left(\frac{1}{\binom{N}{2}} \sum_{\substack{i \in C(\ell(k)) \\ j \in C(r(k))}} \sqrt{q_{ij}} \right) + \sqrt{1-\pi_k} \left(\frac{1}{\binom{N}{2}} \sum_{\substack{i \in C(\ell(k)) \\ j \in C(r(k))}} \sqrt{1-q_{ij}} \right) \right] \tag{6}$$

$$= 1 - \frac{1}{\binom{N}{2}} \sum_m \sum_{\substack{i,j \in C(m) \\ i<j}} \sqrt{q_{ij}} - \sum_k \max_{\pi_k} \left[Q(\ell(k), r(k)) \sqrt{\pi_k} + \tilde{Q}(\ell(k), r(k)) \sqrt{1-\pi_k} \right], \tag{7}$$

where we define

$$Q(s_1, s_2) \equiv \frac{1}{\binom{N}{2}} \sum_{\substack{i \in C(s_1) \\ j \in C(s_2)}} \sqrt{q_{ij}}, \text{ and } \tilde{Q}(s_1, s_2) \equiv \frac{1}{\binom{N}{2}} \sum_{\substack{i \in C(s_1) \\ j \in C(s_2)}} \sqrt{1-q_{ij}}. \tag{8}$$

The value π_k^* that minimizes Equation 7 is

$$\pi_k^* = \frac{Q(\ell(k), r(k))^2}{Q(\ell(k), r(k))^2 + \tilde{Q}(\ell(k), r(k))^2}, \tag{9}$$

and the term of the summation involving k resulting from substituting in π_k^* becomes

$$\max_{\pi_k} \left[Q(\ell(k), r(k))\sqrt{\pi_k} + \tilde{Q}(\ell(k), r(k))\sqrt{1-\pi_k} \right] = \sqrt{Q(\ell(k), r(k))^2 + \tilde{Q}(\ell(k), r(k))^2} \tag{10}$$

$$= \|\boldsymbol{z}(\ell(k), r(k))\|_2, \tag{11}$$

where $\boldsymbol{z}(s_1, s_2) \in \mathbb{R}^2$ and is defined to be

$$\boldsymbol{z}(s_1, s_2) \equiv [Q(s_1, s_2), \tilde{Q}(s_1, s_2)]. \tag{12}$$

A.2.2 Effect of a Swap on Minimum Achievable Hellinger Distance

Most terms in the expression for $D(\pi, q)$ from Equation 5 will not change as a result of a swap of s_1 and s_2. The terms that do change are those corresponding to nodes along the path from s_1 to s_2, not including s_1 and s_2 themselves. Let k be a node along the path from s_1 to $LCA(s_1, s_2)$, not including the endpoints. Define $u(k)$ to be the child of k that is not an ancestor of s_1, and $g(k)$ to be the child of k that is an ancestor of s_1. Then the decrease in $D(\pi, q)$ due to k from making the swap is:

$$\Delta_k(s_1, s_2) = \|\boldsymbol{z}(g(k) \setminus s_1, u(k)) + \boldsymbol{z}(s_2, u(k))\| - \|\boldsymbol{z}(g(k) \setminus s_1, u(k)) + \boldsymbol{z}(s_1, u(k))\| \tag{13}$$

The change in $D(\pi, q)$ due to a node along the path from s_2 to $LCA(s_1, s_2)$ is similar to Equation 13 but with the roles of s_1 and s_2 reversed.

At the $LCA(s_1, s_2)$ the change is

$$\begin{aligned}\Delta_{LCA}(s_1, s_2) =& \|\boldsymbol{z}(g(LCA) \setminus s_1, h(LCA) \setminus s_2) + \boldsymbol{z}(s_1, s_2) + \boldsymbol{z}(g(LCA) \setminus s_1, s_1) + \boldsymbol{z}(s_2, h(LCA) \setminus s_2)\| - \\ & \|\boldsymbol{z}(g(LCA) \setminus s_1, h(LCA) \setminus s_2) + \boldsymbol{z}(s_1, s_2) + \boldsymbol{z}(g(LCA) \setminus s_1, s_2) + \boldsymbol{z}(s1, h(LCA) \setminus s_2)\|,\end{aligned} \tag{14}$$

where LCA is used as shorthand for $LCA(s_1, s_2)$. The total change in $D(\pi, q)$ is then

$$\sum_{k \in path(s_1, LCA(s_1, s_2))} \Delta_k(s_1, s_2) + \sum_{k \in path(s_2, LCA(s_1, s_2))} \Delta_k(s_2, s_1) + \Delta_{LCA}(s_1, s_2). \tag{15}$$

A.2.3 Approximating the Effect of a Swap

The quantity in Equation 15 depends on the values of q_{ij} for the relevant pixel pairs, which will be unknown at test time. We note that from Equations 9 and 12 that

$$\boldsymbol{z}(s_1, s_2) = \|\boldsymbol{z}(s_1, s_2)\|_2 [\sqrt{\pi_k^*}, \sqrt{1-\pi_k^*}]. \tag{16}$$

We choose to bound this based on the product of the size of the regions:

$$\|\boldsymbol{z}(s_1, s_2)\|_2 \leq \frac{|C(s_1)||C(s_2)|}{\binom{N}{2}}, \tag{17}$$

with equality iff either $q_{ij} = 1$ for all $i \in C(s_1), j \in C(s_2)$ or $q_{ij} = 0$ for all $i \in C(s_1), j \in C(s_2)$. If we have an estimate $\hat{\pi}(s_1, s_2)$ of any two regions s_1 and s_2, we can combine Equations 16 and 17 to obtain an estimate for $\boldsymbol{z}(s_1, s_2)$:

$$\hat{\boldsymbol{z}}(s_1, s_2) \approx \frac{|C(s_1)||C(s_2)|}{\binom{N}{2}}[\sqrt{\hat{\pi}(s_1, s_2)}, \sqrt{1 - \hat{\pi}(s_1, s_2)}]. \tag{18}$$

At test time, we can therefore use model predictions of $\boldsymbol{\pi}$ to estimate the change in Hellinger distance by plugging the approximate $\hat{\boldsymbol{z}}$ values from Equation 18 into the change in Hellinger distance from Equation 15.

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 ADDITIONAL DATASET DETAILS

We generated four synthetic ground truth segmentations per example in the Penn-Fudan pedestrian dataset, each of which represents a different prototypical parse of the image. The original semantic segmentation labels contained thirteen classes: background, hair, face, upper and lower clothes, and left and right arms, hands, legs, and shoes. The first synthetic ground truth treats each of these semantic classes as a separate region. The second merges hair with face, hand with corresponding arm, and shoe with corresponding leg. The third further merges all lower body classes, upper body classes, and head classes together. The fourth and final ground truth segmentation distinguishes only between background and non-background semantic classes. When forming the ground truths, regions were split as necessary to ensure that each is spatially contiguous. Visualizations of several ground truths and corresponding images are shown in Figure 9.

B.2 ADDITIONAL SST TRAINING DETAILS

Our network for estimating marginal probabilities used two sets of features as input. The first are visual features extracted from VGG-19 (Simonyan & Zisserman, 2014). Each image was run through the net, and the activations just prior to the second max-pooling operation were recorded (fourth convolutional layer). These activations were then upsampled back to the original image size using nearest neighbor interpolation, thereby producing 128 features for each pixel. These features were average-pooled within the base-level regions to give 128 visual features for each. The second set were geometric features regarding the regions and their relations. These features included: centroids of the parent, left, and right regions; difference and absolute difference between the left and right centroids; coordinates of the top left and bottom right bounding box corners of each region; height and width of the regions; normalized number of pixels of each region (maximum of 1); the difference and absolute difference of the normalized number of pixels in the child regions; and the difference between the parent normalized number of pixels and minimum (respectively maximum) of child normalized number of pixels. All coordinates were expressed relative to the image size such that both x and y coordinates ranged from -1 to 1. This resulted in 35 additional geometric features, yield 163 features total.

Each set of features has the benefit that the features of a parent region can efficiently be computed given features of the child regions. All features were normalized to have zero mean and unit variance across the training examples.

For BSDS500, the first layer of the marginal estimation network had 128 hidden units and the second layer had 64. For Penn-Fudan the first layer had 32 hidden units and the second had 16. ReLU activations with dropout were used for the first and second layers. The final layer had a single sigmoid output and no dropout.

Figure 9: Example synthetic ground truths created for the Penn-Fudan dataset. From left to right: image, semantic segmentation labels, ground truths 1-4. The colors shown in the ground truths are not semantically meaningful: they are chosen to simply be maximally distinguishable for ease of visualization.

B.3 ADDITIONAL DETAILS REGARDING BOYKOV-JOLLY BASELINE

In BOYKOV-JOLLY, a strength is computed between any two adjacent base-level regions s_1 and s_2 as the mean of $1 - \frac{\exp(-(I_i - I_j)^2)}{2\sigma^2}$, where pixel $i \in s_1$ and pixel $j \in s_2$. Here $I_i \in [0, 1]$ denotes the mean intensity at pixel i across color channels. Merges are then sequentially made between regions in order of ascending strength. The strength between two arbitrary regions r_1 and r_2 is defined to be the minimum superpixel strength between two adjacent superpixels s_1 and s_2 such that $s_1 \in r_1$ and $s_2 \in r_2$. Merges continue in this way until a single region remains. This sequence of merges induces a segmentation tree, which can be thresholded by any strength from 0 to 1 in order to form a segmentation. In the results presented in Tables 1 and 2 of the main paper, a single threshold and σ with the best validation performance were chosen via grid search.

B.4 ADDITIONAL SAMPLES

We show additional samples from the SST-UTREE algorithm trained on BSDS 500 in Figure 10 and representative samples from SST-UTREE on Penn-Fudan in Figure 11.

Figure 10: Samples from the SST-UTREE model on BSDS500. From left to right: image, samples 1-4 from the model. All images are from the test set.

Figure 11: Representative samples from the SST-UTREE model on Penn-Fudan. Within each pane, from left to right: image, samples 1-2 from the corresponding test model. Note that for visualization purposes, the images and segmentations have been resized to the same dimensions; in the actual dataset the sizes vary.