Discovering Efficiency in Coarse-To-Fine Texture Classification

Jonathan Barron
Jitendra Malik

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-94
http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-94.html

June 12, 2010

Discovering Efficiency in Coarse-To-Fine Texture Classification

Jonathan T. Barron and Jitendra Malik
Computer Science Division
University of California at Berkeley, Berkeley, CA, 94720
{barron, malik}@eecs.berkeley.edu

Abstract

We introduce a model for joint texture classification and segmentation that learns not only **how** *to classify accurately, but* **when** *to classify efficiently. This model, combined with a complementary efficient feature representation that we describe, allows us to move beyond naive sliding-window classification strategies into sub-linear coarse-to-fine classification of an entire image. Recognition is formulated as a scale-space traversal through the image in which we can "stop short" at coarse scales, dramatically increasing both the speed and the accuracy of classification. Unlike other models, ours is constructed such that the classification produced when stopping-short is exact (that is, equivalent to the classification produced when not stopping-short), because coarse-to-fine efficiency is directly incorporated into the model. Classification is demonstrated on partially- and fully-annotated datasets of satellite and medical imagery.*

1. Introduction

The coarse-to-fine paradigm has long an important part of efficient classification. Much work has used this approach for efficient object detection within local windows, through techniques such as boosted decision trees [1] or hierarchical decompositions of objects [2]. More recent work [3] has used branch-and-bound for efficient sub-window object localization. We will build on this tradition to develop an approach towards parsing the entirety of an image, in which coarse-to-fine efficiency comes not from how we classify a single sub-region of an image, but from how the classification of an entire image proceeds — and when it is terminated.

We will construct a model for efficient recognition and segmentation within the framework of a hierarchical scale-space traversal of an image. Building on previous work [4] in maximum entropy Markov models (MEMMs), we will introduce a novel model that learns how to terminate the traversal of an image at coarse scales while still producing the same results as a complete traversal. This allows for accurate classification that is, on average, of sub-linear complexity relative to the size of the image. A quad-tree[5] is used as a medium for multiresolution inference, and we features inspired by the integral image technique in [1] for efficient feature generation. The output of this system is shown in Figure 1, and a cartoon depiction of classification is shown in Figure 2.

Multiresolution models have long been used for compact image representation[6], motion estimation[7], as well as classification and segmentation[8]. In [9], multiscale random fields are used for belief propagation through a quad-tree hierarchy over an image. In [10], mixtures of tree-structured belief networks are used for structural scene de-

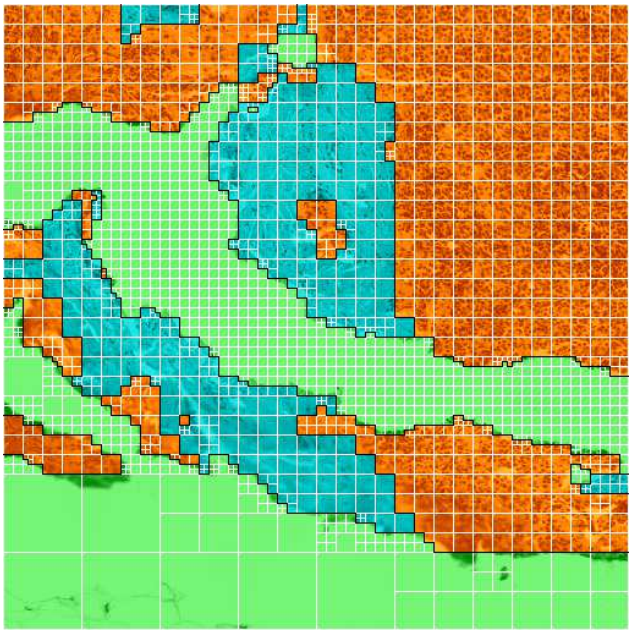


Figure 1. The output of our classifier ($T = 8$, $\epsilon_\emptyset = 0.25$, overlapping windows) on a test image from our "Tumor" dataset. Luminance is from the input image and colors are from the classifier's annotation. Boxes show what regions are observed, and therefore denote the scale at which inference is terminated. Black edges separate different classes, white edges do not.

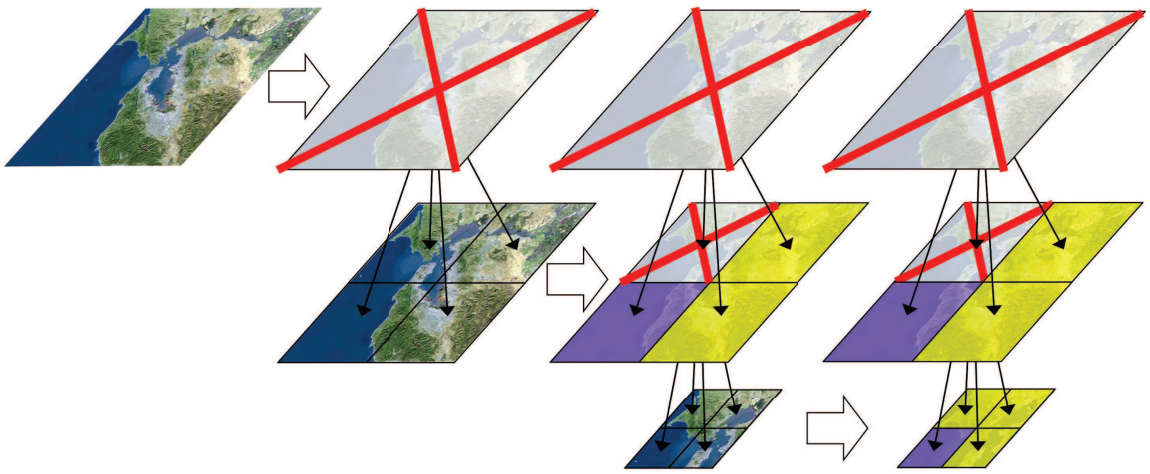


Figure 2. Efficient classification in our model. We observe the image and then either: label and terminate, or subdivide and recurse. Branches of the quad-tree are generated as required. Colors represent label assignments, red X's represent subdivisions, and arrows indicate steps through scale.

composition. More recent work such as [11] uses hierarchical Dirichlet process hidden Markov trees to model natural scenes for the purpose of scene recognition. These works all have a similar hierarchical motivation as ours, but we have the fundamentally different goal of directly addressing efficient inference.

Maximum entropy Markov models have long been used in natural language processing for topic estimation[4] and document segmentation[12]. We will consider sequences in scale, rather than in time. We are not the first to use such a framework for computer vision applications, though most such work is based on conditional random fields (CRFs)[13], a successor to MEMMs. In [14] multiscale CRFs are used to improve local texture classifiers, and in [15] hierarchical CRFs are used for contextual image labeling and object recognition. Though this work shares similarities with ours, we are primarily interested in efficient classification, not in modeling context. Additionally, a CRF-style model seems poorly-suited to our goal, as we will explain later.

2. Features

Our feature extraction pipeline is shown in Figure 3. We first separate an image into a set of C grayscale channels (we use the channels of the YUV colorspace) and then construct an integral histogram[16] for each channel. The B-sized vector (where B is the number of histogram bins, indexed by b) at location x, y of channel c's integral histogram $H^c(x, y, b)$ contains the sum of all other vectors above and to the left of x, y. These integral histograms can be "queried" similarly to an integral image [17][1], but instead of returning the sum of all values in channel c within this region, this operation returns a b dimensional histogram of the values. We concatenate the histograms from all C channels to make a D-dimensional feature vector for a region ($D = BC$). Constructing and querying integral histograms are extremely fast — $\mathcal{O}(NC)$ and $\mathcal{O}(CB)$ respectively, where N is the number of pixels in the image. Construction requires no more computations than does naively computing histograms for the entire image, though significantly more memory is required to store the integral histogram. This means that after minimal preprocessing, we can generate the intensity/color histogram of a rectangular image region quickly, regardless of its size. Without this optimization, we would be forced to repeatedly generate histograms of overlapping regions — an expensive proposition.

The histogram binning thresholds are set such that the distributions of the histograms in our training set are uniform. In our tests we set $B = 30$, though smaller values are often just as effective.

These integral histograms let us rapidly construct a quad-tree of the image. We query the rectangular region that is the entire image, subdivide that region into four same-sized sub-regions, and repeat until we have a full quad-tree of some maximum depth T, composed of color/intensity histogram features. At test time, the quad-tree is constructed as-needed, and we only subdivide a region if our trained model requires it (and if we have not yet reached depth T).

Our feature representation and extraction method can easily be extended to richer features such as HOG descriptors[18] or textons[19]. Similar to [20], we can even integrate a rigid-object detector into our model. We need only produce additional channels, such as maps of vector-quantized texton indices or scored object detections, and use these channels in the rest of the pipeline. This increases accuracy, but generating elaborate features necessarily slows

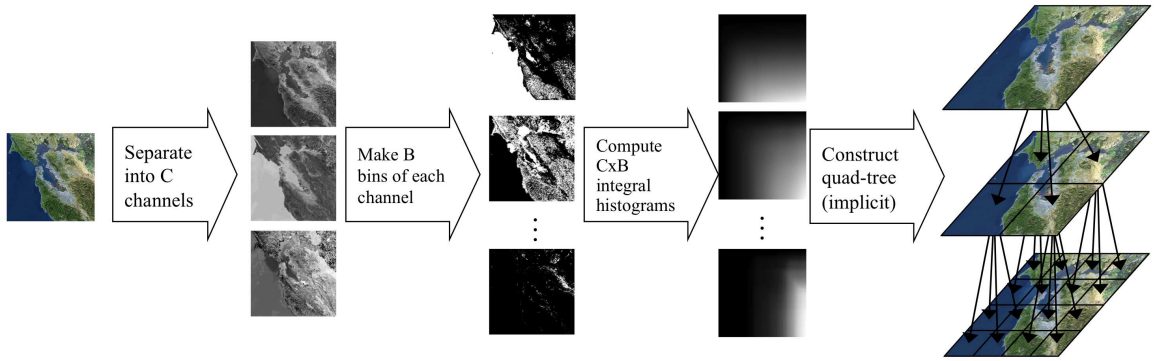


Figure 3. The feature extraction pipeline. We separate the image into grayscale channels (in our case, the YUV colorspace), and construct integral histograms for each channel. This is equivalent to (but faster than) binning each channel and integrating the resulting masks. These integral histograms can be used during inference to rapidly construct a quad-tree, on-demand.

classification.

3. Model

In Figure 2 classification is shown as a traversal through the quad-tree from coarse to fine. The classifier either assigns a label and terminates, or sub-divides the image and recursively classifies its four children. We will construct a model that assigns measures of "confidence" to labels while traversing the tree, and during classification our model will use that confidence to make decisions regarding label-assignment and recursion. We use this non-standard terminology because the intermediate "confidence" distributions over labels do not resemble the output of standard statistical models (as the confidence over labels only increases). The internal structure of our model should be thought of as a stochastic automaton that progressively construct a single distribution over labels at the leaves of the quad-tree.

Consider the model in Figure 4. For the K labels in our dataset, we construct K "label-states". We define $\alpha_t(s_k)$ as the confidence our model has in label k at depth t. We construct an additional state, the "active-state" s_a, where all confidence exists before being assigned to the label states. Effectively, $\alpha_t(s_a)$ measures the *lack* of confidence in the label-states at depth t. $P_{s_a}(s|o_t)$ determines how much confidence is assigned to the label-states and how much stays put, based on o_t, the region of the quad-tree being observed. Confidence is *permanently* assigned to the label-states — confidence in the active-state can only decrease and confidence in a label-state can only increase. At each depth, $P_{s_a}(s|o_t)$ and the α's and are normalized and are between 0 to 1, making this model a stochastic automaton.

A MEMM is a variant of a hidden Markov model (HMM), where we condition on the observed data instead building a joint model of observations and states. This is analogous to the difference between a Markov random field (MRF) and a CRF. Normally, the hidden states in a MEMM are fully connected and completely equivalent. We will instead use this stochastic automaton as our hidden states, dramatically restricting the model. The top-down traversal through an image pyramid can be thought of as many separate observation sequences (one of which is rendered in red in Figure 5). In our notation we will first consider only one of these sequences. We will call an MEMM with this special structure imposed on the hidden states a "spatial" MEMM (S-MEMM). A S-MEMM models the assumption that anything we believe about a region we must also believe about any subset of that region, which is why confidence is permanently allocated to labels.

In standard MEMM notation (similar to that of [12]), belief propagation is defined as follows:

$$\alpha_{t+1}(s) = \sum_{s' \in S} \alpha_t(s') P_{s'}(s|o_t) \tag{1}$$

Where $\alpha_t(s)$ is the probability of being in state s at time t given a sequence of observations $\langle o_1, o_2, \cdots, o_t \rangle$, and S is the set of all hidden states. $\alpha_t(s)$ still corresponds to confidence, and o_t is still the region of the quad-tree at depth t in our sequence. We will now revise this model to reflect the previously-described restrictions placed on the hidden states in our stochastic automaton.

$$\alpha_0(s) = \begin{cases} 1 & s = s_a \\ 0 & o.w. \end{cases} \tag{2}$$

$$\alpha_{t+1}(s) = \begin{cases} \alpha_t(s_a) P_{s_a}(s|o_t) & s = s_a \\ \alpha_t(s_a) P_{s_a}(s|o_t) + \alpha_t(s) & o.w. \end{cases} \tag{3}$$

Additionally, we fix $P_{s_a}(s_a|o_T) = 0$, which means that at the leaves of the quad-tree the active-state has no confidence and the alpha values over just the label states are normalized.

We now define the transition probabilities for the active-state:

$$P_{s_a}(s|o_t) = \frac{1}{Z(t)} \exp\left(\vec{w}_{st} \cdot \vec{f}(o_t)\right) \tag{4}$$

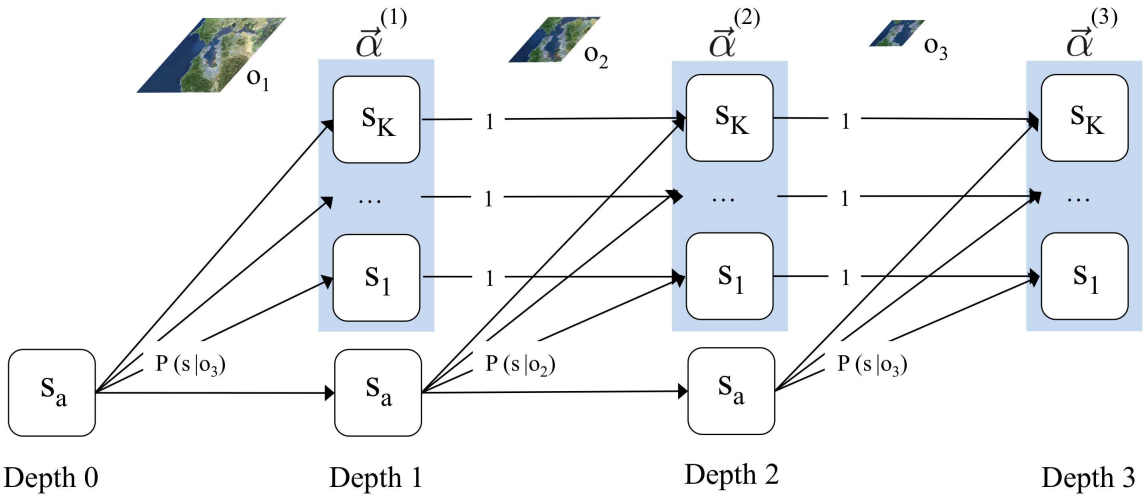


Figure 4. Our model, shown as a HMM / CRF trellis — **not** as a graphical model. All confidence initially sits at the active-state s_a at the top of the quad-tree. As we traverse the tree from coarse to fine, that confidence either stays put or is permanently assigned to the label-states (highlighted). This continues until the maximum depth T. The transition probabilities $P_{s_a}(s|o_t)$ are generated from o_t, the observed sub-region at depth t. This is one of many sequences through the quad-tree, which we formally describe separately.

This is the standard maximum-entropy classifier, or the "softmax" activation function in neural-network nomenclature. $\vec{f}(o_t)$ is the D-dimensional feature vector we have for observation o_t (the intensity/color histograms computed by querying $H^c(i,j,d)$ for all C channels with the bounding box of o_t), and $\vec{w}_{st}$ is the D-dimensional vector of corresponding weights for state s, at scale t. $Z(t)$ is a normalizing term.

Our quad-tree has $N = 4^{T-1}$ leaves. We represent the S-MEMM's predicted distributions over label-states for each leaf with an α vector:

$$\vec{\alpha}_T^{(i)} = \left[\alpha_T^{(i)}(s_1),\ \alpha_T^{(i)}(s_2),\ \cdots,\ \alpha_T^{(i)}(s_K)\right]^{\mathrm{T}} \tag{5}$$

Here we have indexed the N leaves of the quad-tree with (i). Because we fixed $P_{s_a}(s_a|o_T) = 0$, these vectors are normalized even though we do not include $\alpha_T(s_a)$. From our ground-truth annotation we can construct N corresponding label vectors:

$$\vec{\ell}_T^{(i)} = \left[\ell_T^{(i)}(s_1),\ \ell_T^{(i)}(s_2),\ \cdots,\ \ell_T^{(i)}(s_K)\right]^{\mathrm{T}} \tag{6}$$

Where $\ell_T^{(i)}(s_k)$ is the fraction of pixels in leaf i in the ground-truth annotation that are labeled k. Because some pixels may be unlabeled, these $\vec{\ell}_T^{(i)}$ vectors may not be normalized.

4. Training

Though we needed to describe the internal workings of the S-MEMM with non-traditional terms (allocating confidence, etc), at this point we can effectively ignore this internal structure and consider the model as simply producing a set of distributions over labels at the leaves of the quad-tree, given a quad-tree. The $\vec{\alpha}_T^{(i)}$ vectors are the predicted label distributions, and the $\vec{\ell}_T^{(i)}$ vectors are the ground-truth label distributions whose likelihood we wish to maximize. Minimizing KL-divergence would be a sensible way to minimize the difference between the two, but should not be used as, $\vec{\ell}_T^{(i)}$ may not be normalized. We therefore construct what we call "partial" KL-divergence:

$$D_{pKL}\left(\vec{\ell}_T^{(i)}||\vec{\alpha}_T^{(i)}\right) = \sum_{k=1}^{K} \ell_T^{(i)}(s_k) \log\left(\max\left(1, \frac{\ell_T^{(i)}(s_k)}{\alpha_T^{(i)}(s_k)}\right)\right) \tag{7}$$

Partial KL-divergence only penalizes *under-predicting* the label distribution, and does not penalize the model for *over-predicting* any label. In practice, partial KL-divergence behaves nearly identically to KL-divergence when our annotation is complete, while giving us robustness to partial annotations.

Our loss function is the sum of the partial KL-divergence of each $\vec{\alpha}$-$\vec{\ell}$ pair, plus a regularizing term:

$$\mathcal{L} = \sum_{i=1}^{N}\left(D_{pKL}\left(\vec{\ell}_T^{(i)}||\vec{\alpha}_T^{(i)}\right)\right) + \lambda\sqrt{\sum_{t=1}^{T}\left(||\vec{w}_{st}||_2^2\right)^2} \tag{8}$$

Rather than having a single Gaussian prior over the weights at all scales, we take the L^2 norm of the sum of squared weights at each scale. The former strategy encourages all of the weights at certain scales to be set to zero, while our strategy evenly penalizes the weights at each scale. This prevents overfitting while allowing all scales of the S-MEMM to contribute to classification.

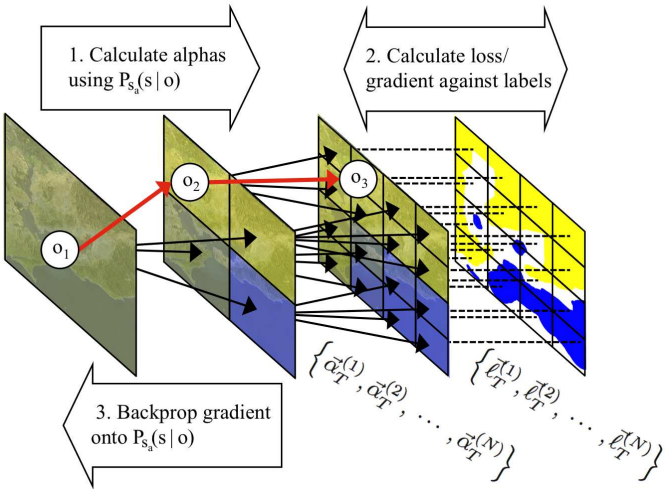


Figure 5. The learning procedure for a single image. The traversal through the quad-tree is decomposed into a series of sequences, one of which is highlighted in red. We do belief propagation down to the leaves of the quad-tree, and compare the predicted label distributions to the ground-truth label distributions with partial KL-divergence. The gradient of the divergence is backpropagated up through the quad-tree, and then onto the weights of $P_{s_a}(s|o_t)$.

Minimizing partial KL-divergence implicitly optimizes for efficiency by encouraging the assignment of confidence at coarse scales to prevent errors at fine scales. This coincides with "label bias" [13], which describes the MEMM's tendency to favor states with few outgoing transitions because of the per-state normalization of transition probabilities. In most uses of MEMMs, this is a *problem*, but here it is exactly what we want. Our label-states have only one outgoing transition, so our model favors them over the active-state, thereby favoring classification at coarse scales. Since CRFs are meant to prevent label bias, they should be poorly-suited to this sort of learned efficiency.

To encourage efficiency when it is *not* necessary for accuracy, we can impose a fixed bias on the transition probabilities from the active-state to the label-states, which dampens the transitions probabilities to the active-state:

$$P_{s_a}(s|o_t) = \begin{cases} \frac{1-\beta(t)}{Z(t)} \exp(\vec{w}_s \cdot \vec{f}(o_t)) & s = s_a \\ \frac{1-\beta(t)}{Z(t)} \exp(\vec{w}_s \cdot \vec{f}(o_t)) + \frac{\beta(t)}{K} & o.w. \end{cases} \quad (9)$$

$$\beta(t) = (\epsilon_\emptyset)\, 4^{t-T} \quad (10)$$

Where $\beta(t)$ is a bias towards label-states at height t, which is weighted by $\epsilon_\emptyset$. $\epsilon_\emptyset$ can range from 0 (no bias) to 1 (where the transition probabilities at the leaves are completely defined by the bias). The 4^{t-T} multiplier reflects the exponential growth in the number of observations as t grows, and accordingly adjusts the transition probabilities such that the total number of observations required is evenly minimized. During learning, the model compensates for this bias by assigning confidence to the label states at more coarse scales. This term can be thought of as an "efficiency bias", a method for biasing the model towards efficiency.

We optimize $\mathcal{L}$ using the conjugate gradient method. This requires efficient calculation of $\mathcal{L}$ and $\partial\mathcal{L}/\partial\vec{w}_{st}$, both of which can be done with dynamic programming, in the manner of the forward part of the forward-backward algorithm [21]. Learning is demonstrated in Figure 5. We set λ to maximize accuracy on the validation set. The $\epsilon_\emptyset$ parameter should be considered "user-defined", and can be set according to the desired tradeoff of accuracy for efficiency. T can be selected a-priori based on the desired granularity of the segmentation — though we will demonstrate that our model is inherently robust to T being too large.

Training a S-MEMM on 8 training images ($T = 7$) usually takes about 20 minutes on a 2008 Macbook Pro. Though not necessary, it is faster to first learn one set of weights for all scales, and use those weights to initialize the weights at each scale.

5. Inference

Once we have learned a set of weights, we can infer the distribution of label-states at the leaves of the quad-tree by calculating all $\vec{\alpha}_T^{(i)}$. We choose $\hat{s}^{(i)}$ to be the label with the highest probability:

$$\hat{s}^{(i)} = \arg\max_s \vec{\alpha}_T^{(i)}. \quad (11)$$

The fundamental strength of the S-MEMM is that we *need not* always compute $\vec{\alpha}_T^{(i)}$ to exactly determine $\hat{s}^{(i)}$. By the construction of our model, we know that:

$$\alpha_t^{(i)}(s) \le \alpha_{t+1}^{(i)}(s) \le \alpha_t^{(i)}(s) + \alpha_t^{(i)}(s_a) \quad \forall s \neq s_a \quad (12)$$

$$\alpha_t^{(i)}(s_a) \ge \alpha_{t+1}^{(i)}(s_a). \quad (13)$$

If we assume that

$$\alpha_t^{(i)}(\hat{s}) > \max_{s' \neq \{\hat{s}, s_a\}} \left(\alpha_t^{(i)}(s')\right) + \alpha_t^{(i)}(s_a) \quad (14)$$

then by the previously stated inequalities, it must follow that

$$\alpha_{t+1}^{(i)}(\hat{s}) > \max_{s' \neq \{\hat{s}, s_a\}} \left(\alpha_{t+1}^{(i)}(s')\right) + \alpha_{t+1}^{(i)}(s_a) \quad (15)$$

and therefore, by induction, we have

$$\alpha_T^{(i)}(\hat{s}) > \max_{s' \neq \{\hat{s}, s_a\}} \left(\alpha_T^{(i)}(s')\right) + \alpha_T^{(i)}(s_a) \quad (16)$$

which is equivalent to

$$\hat{s}^{(i)} = \arg\max_s \vec{\alpha}_T^{(i)}. \quad (17)$$

What we have demonstrated is that, at any time in inference, if the difference between the confidence of the most-likely

"Tumor" dataset

	T = 4	T = 5	T = 6	T = 7	T = 8	T = 9
Naive	83.3% / 64	88.5% / 256	90.4% / 1024	90.0% / 4096	88.4% / 16384	86.8% / 65536
S-MEMM ($\epsilon_\emptyset = 0$)	**83.7%** / 81	**88.8%** / 310	**91.1%** / 1220	**91.6%** / 1719	**91.9%** / 2742	**92.0%** / 4193
S-MEMM ($\epsilon_\emptyset = 0.1$)	83.6% / 81	88.5% / 310	90.5% / 635	91.6% / 1271	91.8% / 2145	91.9% / 3190
S-MEMM ($\epsilon_\emptyset = 0.25$)	80.4% / 71	85.8% / 170	89.3% / 444	90.7% / 767	91.7% / 1731	91.9% / 2426
Naive (overlap)	81.3% / 64	86.7% / 256	90.3% / 1024	91.5% / 4096	90.5% / 16384	88.6% / 65536
S-MEMM (overlap, $\epsilon_\emptyset = 0$)	82.3% / 84	87.5% / 321	**91.5%** / 1244	**92.6%** / 3697	**92.9%** / 5295	**93.1%** / 9207
S-MEMM (overlap, $\epsilon_\emptyset = 0.1$)	**83.6%** / 81	**88.5%** / 309	90.4% / 626	91.6% / 1268	91.8% / 2147	92.0% / 3136
S-MEMM (overlap, $\epsilon_\emptyset = 0.25$)	77.7% / 75	83.9% / 204	88.1% / 487	91.4% / 1437	92.2% / 1972	91.9% / 2404

"Gmaps" dataset

	T = 4	T = 5	T = 6	T = 7	T = 8	T = 9
Naive	**81.0%** / 64	**82.2%** / 256	82.1% / 1024	81.4% / 4096	80.5% / 16384	79.4% / 65536
S-MEMM ($\epsilon_\emptyset = 0$)	78.4% / 76	81.3% / 261	**83.7%** / 622	**84.4%** / 1358	83.8% / 2003	**83.8%** / 1972
S-MEMM ($\epsilon_\emptyset = 0.1$)	80.2% / 73	81.6% / 176	82.8% / 464	83.1% / 761	**84.0%** / 1518	83.8% / 1844
S-MEMM ($\epsilon_\emptyset = 0.25$)	78.8% / 49	80.7% / 126	81.6% / 259	83.0% / 597	83.2% / 834	83.7% / 1736
Naive (overlap)	**79.1%** / 64	**82.7%** / 256	83.1% / 1024	82.8% / 4096	81.8% / 16384	80.6% / 65536
S-MEMM (overlap, $\epsilon_\emptyset = 0$)	78.9% / 85	81.0% / 301	**83.5%** / 881	**84.0%** / 1510	**84.1%** / 5359	**84.0%** / 6015
S-MEMM (overlap, $\epsilon_\emptyset = 0.1$)	78.7% / 78	81.7% / 209	83.1% / 534	83.6% / 1198	83.6% / 2432	83.8% / 4962
S-MEMM (overlap, $\epsilon_\emptyset = 0.25$)	76.4% / 52	79.2% / 130	81.0% / 339	83.0% / 672	83.6% / 1488	83.7% / 2332

Table 1. A comparison of our model against a naive model, with overlapping and non-overlapping windows, as the maximum quad-tree depth T increases. The format is: "accuracy / number of classifications". Bold-face indicates the most accurate model for that value of T.

label-state and that of the second-most-likely label-state is greater than the confidence of the active-state, immediately terminating inference will still yield the same label estimate that complete inference would. Therefore, we always terminate whenever this condition is satisfied, dramatically increasing the efficiency of classification while never affecting accuracy. This ability to terminate early is a direct consequence of our decision to permanently allocate confidence to the label states. Since this technique requires per-state normalization (the lack of which is a defining feature of CRFs) there appears to be no real analogue for CRFs.

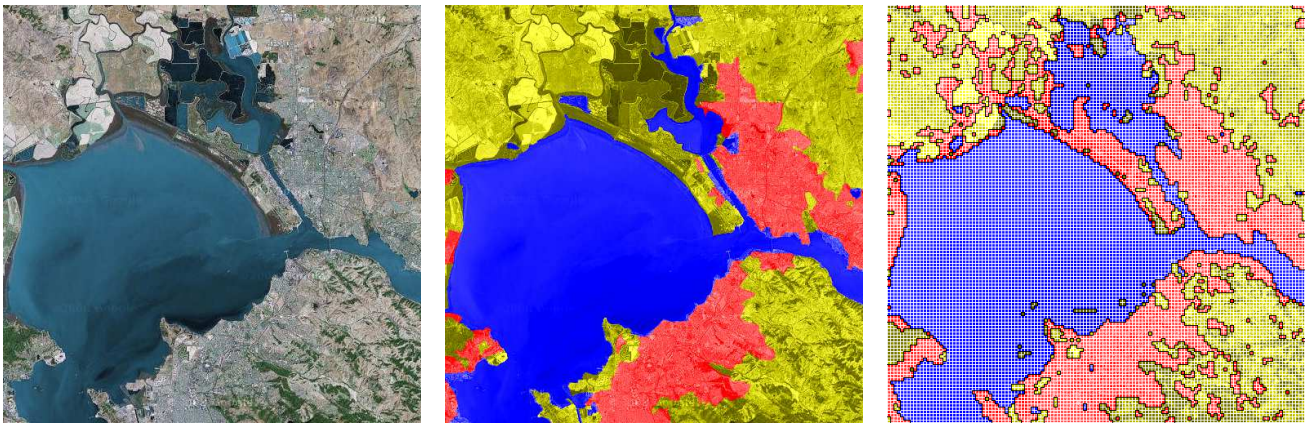
6. Results

We experimented on two new datasets. One is 20 images from an H&E stained histology slide of a tumor taken from [22], and the other is 20 satellite images of earth taken from [23]. Each dataset is divided into 8 training, 4 validation, and 8 test images. The "Tumor" annotations are hand-drawn and partial (some pixels are unlabeled), and labels correspond to "connective tissue", "cancer", and "background". The annotations for the "Gmaps" dataset (see Figure 6(b)) are a function of the Google Maps "map" layer[24], and are therefore complete (every pixel is annotated). The label roughly correspond to "water", "urban", and "park". Images are 512×512 pixels.

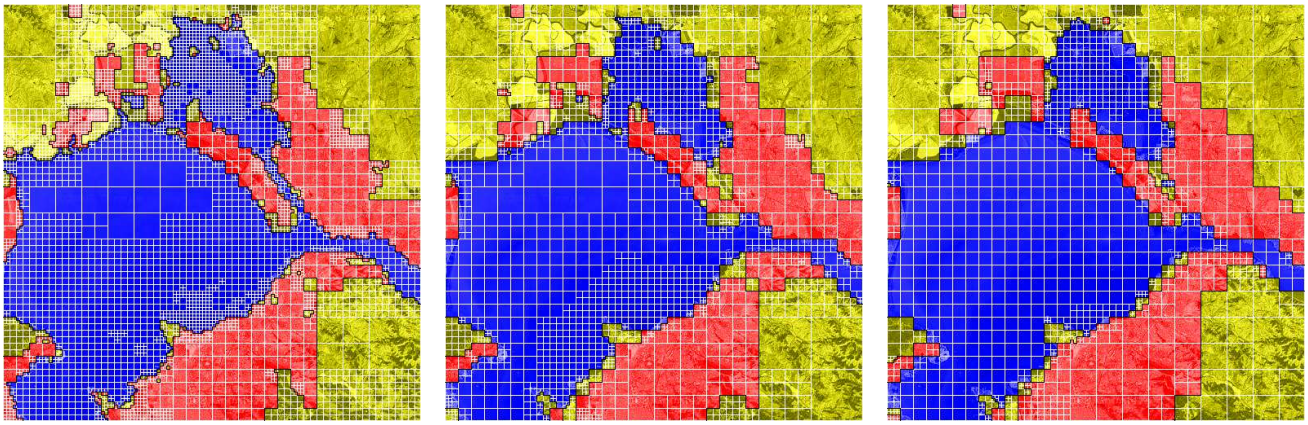
We define *accuracy* as the fraction of labeled pixels that a model classifies correctly. We define a "naive" model, which is a sliding-window model with non-overlapping windows, or a model that only has access to the leaves of the quad-tree (see Figure 6(c)). A model's *efficiency* is the ratio of the number of classifications that a naive model makes to the number that the model makes. The naive model has an efficiency of $1\times$. In this quad-tree framework, the S-MEMM's efficiency has a lower bound of $\frac{3}{4}\times$. For comparison against sliding-window models in which windows overlap, we constructed additional S-MEMM and naive models in which the region used to generate features is twice as large as the region being labeled. The classifications produced by these "overlapping" models (both naive and S-MEMM) appear more "smooth" (labels tend to be contiguous) while having less spatial resolution.

In Table 1, we demonstrate the performance of the S-MEMM against the naive model on both datasets, with overlapping and non-overlapping windows. Figure 6 shows some segmentations produced by these models. As T increases, the naive model's accuracy improves and then worsens, while the S-MEMM's accuracy saturates at a higher value. The S-MEMM is less accurate than the naive model at very small values of T (when neither model performs well), but performs much better at medium and large values of T. When $T = 9$, the S-MEMM is consistently $4-5\%$ more accurate than the naive model, while requiring an order of magnitude fewer classifications. When using overlapping windows this saturation occurs at a larger T, but is at a higher accuracy. This is because overlapping windows make it harder to distinguish textures at coarse scales, but do capture additional information.

Our improved accuracy can be attributed to two causes: The S-MEMM is able to adapt classification to coarse scales (both for efficiency and to avoid mistakes at fine levels), while still using fine scales for a detailed and accurate



(a) A image from the"Gmaps"test-set (b) The image with itsground-truth annotation (c) Naive model: 79.522% / 16384



(d) S-MEMM, $\epsilon_{\emptyset} = 0$:82.322% / 7205 (e) S-MEMM, $\epsilon_{\emptyset} = 0.1$:81.835% / 2993 (f) S-MEMM, $\epsilon_{\emptyset} = 0.25$:81.145% / 2069

Figure 6. Images and classification results ($T = 8$, overlapping windows) for a "Gmaps" test-image. The format is: "accuracy / number of classification". Accuracy and number of classifications are for this image only. Boxes denote observations, and therefore show when inference is terminated.

segmentation. This is why our model asymptotically approaches a high accuracy that the naive model cannot produce. The improved accuracy also comes from how our model is capable of using coarse image patches to bias the inference at finer scales. This can be thought of as a shallow model of context. Both of these causes contribute to the difference between Figures 6(c) and 6(d).

Even without the efficiency bias, the S-MEMM's efficiency is often $10\times$ to $15\times$ that of the naive model. At small values of T the hierarchical model often requires more classifications than the naive model, but efficiency increases dramatically as T grows. Given giga- or tera-pixel images (such as the images that our datasets were taken from) T would probably be in the range of $10 - 15$, and it would not be unreasonable to expect efficiencies on the order of hundreds. The efficiency bias often causes a slight drop in accuracy. Further investigation as to how to encourage efficiency while preserving accuracy is warranted.

7. Conclusion

We have introduced a feature-extraction method, a corresponding scale-space image representation, and a complementary machine learning model for performing exact and efficient classification within this image representation. We have demonstrated how features can be efficiently generated as needed, and how image classification can proceed such that not all features in the scale-space representation of the image need to be observed. We have provided a set of parameters through which the characteristics of classification, in regards to accuracy and efficiency, can be manipulated. The model can easily be extended to richer feature-sets, or more elaborate tree-like image representations, while still being orders of magnitude faster than a model that is ignorant to coarse-to-fine structure. Our model increases *both* the accuracy *and* the efficiency of texture classification, rather than being forced to sacrifice one for the other.

References

[1] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001. 1, 2

[2] Francois Fleuret and Donald Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41(1/2):85–107, 2001. 1

[3] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: object localization by efficient subwindow search. In *CVPR*, 2008. 1

[4] Kamal Nigam, John Lafferty, and Andrew Mccallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999. 1, 2

[5] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, 16(2):187–260, 1984. 1

[6] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31,4:532–540, 1983. 1

[7] James R. Bergen, P. Anandan, Keith J. Hanna, and Rajesh Hingorani. Hierarchical model-based motion estimation. In *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, pages 237–252, London, UK, 1992. Springer-Verlag. 1

[8] Alan S. Willsky. Multiresolution markov models for signal and image processing. In *Proceedings of the IEEE*, pages 1396–1458, 2002. 1

[9] Charles A. Bouman and Michael Shapiro. A multiscale random field model for bayesian image segmentation. *IEEE Transactions on Image Processing*, 3:162–177, 1994. 1

[10] Amos J Storkey and Christopher K I Williams. Dynamic positional trees for structural image analysis. In *In Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pages 298–304. Morgan Kaufmann, 2001. 1

[11] J.J. Kivinen, E.B. Sudderth, and M.I. Jordan. Learning multiscale representations of natural scenes using dirichlet processes. *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct. 2007. 2

[12] Andrew Mccallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proc. 17th International Conf. on Machine Learning*, pages 591–598. Morgan Kaufmann, San Francisco, CA, 2000. 2, 3

[13] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 2, 5

[14] Xuming He, Richard S. Zemel, and Miguel . Carreira-Perpin. Multiscale conditional random fields for image labeling. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:695–702, 2004. 2

[15] Sanjiv Kumar and Martial Hebert. A hierarchical field framework for unified context-based classification. In IEEE, editor, *Tenth IEEE International Conference on Computer Vision (ICCV '05)*, volume 2, pages 1284 – 1291, October 2005. 2

[16] F. Porikli. Integral histogram: a fast way to extract histograms in cartesian spaces. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 829–836 vol. 1, 2005. 2

[17] Patrice Simard, Léon Bottou, Patrick Haffner, and Yann LeCun. Boxlets: a fast convolution algorithm for neural networks and signal processing. In *Advances in Neural Information Processing Systems*, volume 11. MIT Press, Denver, 1999. 2

[18] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005. 2

[19] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int. J. Comput. Vision*, 43(1):29–44, June 2001. 2

[20] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *Proc. 10th European Conference on Computer Vision*, 2008. 2

[21] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989. 5

[22] http://images2.aperio.com/BigTIFF/BreastCancer.tif. 6

[23] http://maps.google.com/?ll=37.791337,-122.348557&t=k&z=11. 6

[24] http://maps.google.com/?ll=37.791337,-122.348557&t=m&z=11. 6