From Label Maps to Label Strokes: Semantic Segmentation for Street Scenes from Incomplete Training Data

Shengqi Zhu Yiqing Yang Li Zhang
University of Wisconsin - Madison
{sqzhu,breakds,lizhang}@cs.wisc.edu

Abstract

This paper proposes a novel image parsing framework to solve the semantic pixel labeling problem from only label strokes. Our framework is based on a network of voters, each of which aggregates both a self voting vector and a neighborhood context. The voters are parameterized using sparse convex coding. To efficiently learn the parameters, we propose a regularized energy function that propagates label information in the training data while taking into account of context interaction and a backward composition algorithm for efficient gradient computation. Our framework is capable of handling label strokes and is scalable to a code book of millions of bases. Our experiment results show the effectiveness of our framework on both synthetic examples and real world applications.

1. Introduction

Image parsing is the task to assign semantic class labels (e.g., tooth, arm, sky, building) to each pixel in the image. It has a variety of real world applications, including parsing street-view images for scene understanding and autonomous vehicle driving.

Many image parsing algorithms, including commonly-used Conditional Random Field (CRF) and exemplar-based methods, require a large set of images with **complete** per-pixel label maps either as training data or as exemplars. Complete label maps are expensive to collect (in terms of time, money, and skill), which limits the robust performance of such methods. Even with the state-of-the-art labeling tool (*e.g.* LabelMe [17]), it still takes over 3 minutes (see task 3 in [19]) to label each image with quasi-complete label maps (only objects of interested are labeled; large regions and background pixels are not labeled).

In this paper, we propose an image parsing model that requires only **strokes**. Figure 1 is an overview of our framework. Our goal is motivated by the success of single image segmentation tools, such as Lazy Snapping [12] and GrabCut [16], where a few strokes are provided to segment foreground and background for image editing applications. These label strokes are cheaper to acquire than (quasi-)complete label maps, hence more scalable. In addition, we do not even require users to label *all* semantic classes in one image; it is sufficient to label only a small *subset* of classes to achieve reasonable parsing performance.

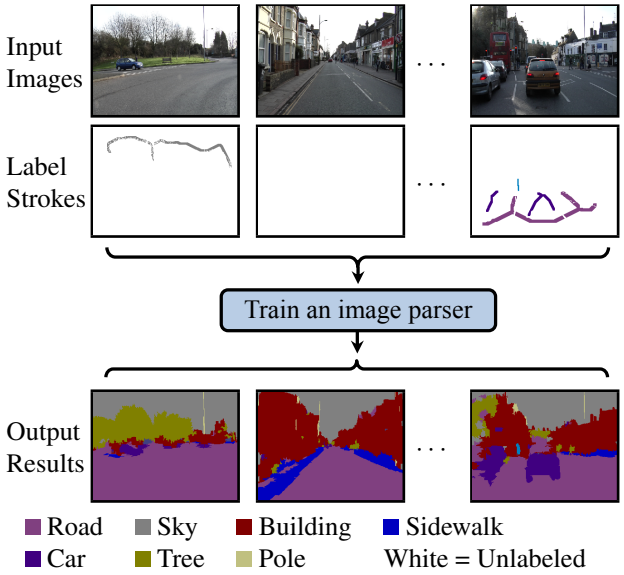


Figure 1: An overview of our image parsing framework. The first row contains raw input images. The second row contains the label strokes for the input images. Each image can have *zero*, one, two, or multiple semantic regions that are labeled by strokes. The output of our framework is an image parsing model as well as a set of full label maps for all the images.

To this end, we formulate semantic image segmentation as a vector-valued function approximation problem. Let $\mathbf{x}$ be an input image. We seek a function Φ that maps $\mathbf{x}$ to an output vector $\mathbf{y} = [y_{i,k}]$ that represents a label probability map, where $y_{i,k}$ is the probability that pixel i belongs to semantic class k. In other words, the output $\mathbf{y}$ is a soft semantic segmentation map. We express the function as

$$\mathbf{y} = \Phi(\mathbf{x}; \Theta) \tag{1}$$

where Θ is the parameter set of the function Φ. This paper is about designing the architecture of Φ and estimating the value of Θ from massive data with tiny supervision.

Technically, we propose a new framework that computes the label probability map of an input image as the equilibrium label distribution on a pixel network of the input image. Our framework is inspired by classic voter models [2],

where each voter (a pixel) both has self opinion and is influenced by its neighbors' opinions. We propose algorithms that automatically construct a pixel network for a given image using a large amount of training images, of which only a small number of strokes are labeled. Our algorithms also efficiently compute the equilibrium label distribution on the pixel network for segmentation. Our contributions include:

- A new model for soft semantic segmentation that outputs pixel label probability.
- A novel algorithm that learns the model parameters from partially labeled stroke images.
- An approach that facilities label propagation in a large image collection while taking into account of context interaction.

2. Related Work

CRF and exemplar-based methods are two well-accepted approaches to semantic image segmentation. Early CRF models use unary and pairwise terms [5, 18], while more recent ones employ high order potentials [8] and long range context interactions [3, 11, 4]. All these CRF methods are fully supervised. More recently, dictionary learning is combined with latent structured-SVM for semantic segmentation [6], in which the latent variables are visual word assignment and fully per-pixel labeled images are assumed as training data. Exemplar-based methods [13, 20, 9, 25] require a large collection of per-pixel labeled images as exemplars. One exception is [14], in which CRF is used to learn segmentation model from partially labeled images.

There are a few other types of weakly-supervised works for semantic image segmentation [21, 22, 23, 24]. The earliest work [21] assumes that an image is associated with a set of words and uses EM-style algorithm to segment the image. More recent work [23, 24] makes a similar input assumption and constructs a multi-image graph with superpixels as nodes for segmentation. Associating many labels to *an entire image* may be too weak to improve the robustness of a segmentation system. Consider street view images, most of them contain similar object types: sky, building, cars, *etc*. For another example, consider face images, most of them have mouths, eyes. Associating a few labels to *a very small number of pixels* may be a more informative form of supervision for training an image parser.

Our work is also related to the recent development of parameter-rich CRF [7, 15], in which decision trees are used to map local features to local potential energy for discrete CRF [7] and continuous CRF [15]. These methods have not yet been used in semantic segmentation, nor have they been shown to deal with partial labels in training data.

Our work is much inspired by [1]. By adding novel constraints and simplifying their formulation, we propose a model that has probabilistic interpretation for both model parameters and model output. More importantly, we propose algorithms that can handle significant amounts of missing labels in training data while [1] is a fully supervised method. Furthermore, our algorithm is scalable to millions of "quantized" states while [1] is limited to a few hundreds. As a result, [1] has been used to post-process an initial semantic segmentation rather than producing one from scratch.

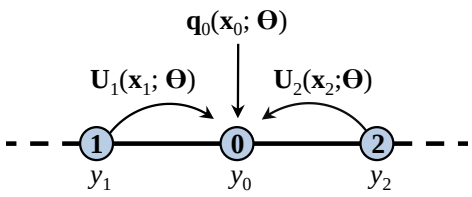


Figure 2: An illustration of a voter in a 3-pixel scenario. To form its label probability y_0, pixel 0 receives both its self vote $\mathbf{q}_0$ as well as propagated votes from its two neighbors via label probability transition matrices $\mathbf{U}_1$ and $\mathbf{U}_2$.

3. A Voter Model for Image Parsing

Given an input image $\mathbf{x}$, we seek to predict its label probability map $\mathbf{y} = [y_{i,k}]$ where $y_{i,k}$ is the probability that pixel i belongs to semantic class k. Denote the label probability vector for pixel i by $y_{i,1:K} = [y_{i,k}]_{k=1}^{K} \in \mathrm{R}^K$. In our model, we assume that $y_{i,1:K}$ is determined by both the feature of pixel i and the context of surrounding pixel label probabilities. We model $\mathbf{y}$ to be the stationary label distribution on a pixel network that is constructed from the input image $\mathbf{x}$.

Specifically, given the input image $\mathbf{x}$, we compute a K-dim **self voting vector** $\mathbf{q}_i(\mathbf{x};\Theta) = [q_{i,k}]_{k=1}^{K} \in \mathrm{R}^K$ for each pixel i based on its own feature, where Θ is the parameter set of our system. Each $q_{i,k}$ is non-negative, representing the self confidence for pixel i belonging to class k; $\sum_{k=1}^{K} q_{i,k}$ is the total label votes from its own feature. We require $\sum_{k=1}^{K} q_{i,k} \leq 1$, so that the remaining label votes for pixel i will come from the pixel labels in its neighborhood. For example, assuming $K = 3$, $\mathbf{q}_i = [0.45; 0; 0.45]$ indicates a strong self vote that pixel i belongs to either class 1 or class 3; $\mathbf{q}_i = [0; 0.1; 0]$ indicates a weak self vote that pixel i belongs to class 2. In both cases, context information needs to be added to the self voting vector $\mathbf{q}_i$ to form the label probability for pixel i.

To model context, let $\mathcal{B}_i$ be the neighborhood of pixel i, which contains the index offsets between pixel i and its neighboring pixels. That is, if $b \in \mathcal{B}_i$, i and $i + b$ are neighbors. For each $b \in \mathcal{B}_i$, we compute a $K \times K$ non-negative **label probability transition matrix** $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$ that propagates $y_{i+b,1:K}$, the label probability vector of pixel $i + b$, to

pixel i as a voting vector $\mathbf{U}_{i,b}(\mathbf{x};\Theta)y_{i+b,1:K}$. As illustrated in Figure 2, we then aggregate the propagated votes with self votes to form the label probability vector for pixel i as:

$$y_{i,1:K} = \mathbf{q}_i(\mathbf{x};\Theta) + \sum_{b\in\mathcal{B}_i} \mathbf{U}_{i,b}(\mathbf{x};\Theta)y_{i+b,1:K} \quad (2)$$

Putting together Eq. (2) for every pixel in the image $\mathbf{x}$ into a big matrix equation, we have

$$\mathbf{y} = \mathbf{h}(\mathbf{x};\Theta) + \mathbf{A}(\mathbf{x};\Theta)\mathbf{y} \quad (3)$$

where $\mathbf{h}(\mathbf{x};\Theta)$ is a vector with $\mathbf{q}_i(\mathbf{x};\Theta)$ as sub-vectors, capturing label votes from self features; $\mathbf{A}(\mathbf{x};\Theta)$ is a matrix with $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$ as sub-matrices, capturing propagated votes due to label interaction between pixels. Eq. (3) defines an equilibrium label distribution $\mathbf{y}$ among all pixels, which we use as the output label probability map, denoted by $\Phi(\mathbf{x};\Theta)$ in Eq. (1). The key question is, given $\mathbf{x}$, how to construct $\mathbf{q}_i(\mathbf{x};\Theta)$ and $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$, so that the stationary label distribution represents a desired soft semantic segmentation. We next discuss (i) when a valid equilibrium label distribution exists, (ii) how to compute the equilibrium distribution, and (iii) how to construct $\mathbf{q}_i(\mathbf{x};\Theta)$ and $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$.

Self-Consistent Constraints We want to ensure that Eq. (2) is *self-consistent*, that is, if $y_{i+b,1:K}$ is a valid probability vector[1] for all $b \in \mathcal{B}_i$, the right hand side of Eq. (2) is guaranteed to be a valid probability vector, regardless of the specific values that each $y_{i+b,1:K}$ takes. Otherwise, a valid equilibrium label distribution may not exist. Specifically, for each pixel i, we enforce the following two constraints on $\mathbf{q}_i$ and $\{\mathbf{U}_{i,b}\}_{b\in\mathcal{B}_i}$:

$$\mathbf{1}^{\mathrm{T}}\mathbf{U}_{i,b} = v_{i,b}\mathbf{1}^{\mathrm{T}} \quad \text{and} \quad \sum_{k=1}^{K} q_{i,k} + \sum_{b\in\mathcal{B}_i} v_{i,b} = 1 \quad (4)$$

The first constraint requires that the sum of elements along each column in $\mathbf{U}_{i,b}$ is the same for all columns, where $v_{i,b}$ represents the total label votes propagated from pixel $i+b$ to pixel i. The second constraint requires that the accumulated label votes for each pixel is 1, including self votes and propagated votes. It is easy to verify that the two constraints in Eq. (4) guarantee Eq. (2) to be self-consistent.

The Equilibrium Solution for $\mathbf{y}$ Eq. (3) can be solved efficiently using Jacobi (or Gauss-Seidel) iteration as

$$\mathbf{y}^{(t)} = \mathbf{h}(\mathbf{x};\Theta) + \mathbf{A}(\mathbf{x};\Theta)\mathbf{y}^{(t-1)} \quad (5)$$

Since Eq. (2) is self-consistent, each iteration produces a valid label probability map. Further, the second constraint in Eq. (4) guarantees that $\sum_{b\in\mathcal{B}_i} v_{i,b} = 1 - \sum_{k=1}^{K} q_{i,k} \leq 1$, which suggests that Eq. (3) is diagonally dominant and therefore Jacobi (or Gauss-Seidel) iteration is guaranteed to converge. We have verified via simulation that the convergence is fast (≤ 10 iterations).

[1] A probability vector has non-negative components that sum up to 1.

Parameterization of $\mathbf{q}_i(\mathbf{x};\Theta)$ and $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$ We formulate $\mathbf{q}_i(\mathbf{x};\Theta)$ and $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$ using sparse convex coding. Without loss of generality, we assume that features extracted at different image locations share the same codebook (bases) for sparse coding, and that the neighborhood for each pixel is the same: $\mathcal{B}_i = \mathcal{B}$ for all pixel i.

Specifically, let $\mathbf{f}_i(\mathbf{x})$ be a local feature extracted from image $\mathbf{x}$ for pixel i. Given $\mathbf{f}_i(\mathbf{x})$, we approximate it using a *sparse convex* combination of bases as

$$\mathbf{f}_i(\mathbf{x}) \longrightarrow \sum_{l=1}^{L} c_{i,l}\mathbf{g}_l \quad (6)$$

where $\{\mathbf{g}_l\}_{l=1}^{L}$ is the **codebook of feature bases**, $\mathbf{c}_i = [c_{i,l}]_{l=1}^{L}$ is the sparse convex **code** for $\mathbf{f}_i(\mathbf{x})$.

To compute $\mathbf{q}_i(\mathbf{x};\Theta)$ given $\mathbf{c}_i$, we associate each basis $\mathbf{g}_l$ in the codebook with a K-dim non-negative vector $\theta_l^{\mathbf{q}} = [\theta_l^{q_k}]_{k=1}^{K}$ and define $\mathbf{q}_i(\mathbf{x};\Theta)$ as a sparse convex combination of $\{\theta_l^{\mathbf{q}}\}_{l=1}^{L}$ using $\mathbf{c}_i(\mathbf{x})$

$$\mathbf{q}_i(\mathbf{x};\Theta) \longleftarrow \sum_{l=1}^{L} c_{i,l}\theta_l^{\mathbf{q}} \quad (7)$$

Similarly, for each $b \in \mathcal{B}$, we associate each basis $\mathbf{g}_l$ in the codebook with a $K \times K$ non-negative matrix $\theta_l^{\mathbf{U}_b}$, and define $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$ as a sparse convex combination of $\{\theta_l^{\mathbf{U}_b}\}_{l=1}^{L}$ using $\mathbf{c}_i(\mathbf{x})$

$$\mathbf{U}_{i,b}(\mathbf{x};\Theta) \longleftarrow \sum_{l=1}^{L} c_{i,l}\theta_l^{\mathbf{U}_b} \quad (8)$$

In Eq. (7) and Eq. (8), $\mathbf{q}_i(\mathbf{x};\Theta)$ and $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$ are defined as convex combinations of $\theta_l^{\mathbf{q}}$ and $\theta_l^{\mathbf{U}_b}$, respectively. With this definition, $\mathbf{q}_i(\mathbf{x};\Theta)$ and $\mathbf{U}_{i,b}(\mathbf{x};\Theta)$ are guaranteed to satisfy the self-consistent constraints in Eq. (4) if $\theta_l^{\mathbf{q}}$ and $\theta_l^{\mathbf{U}_b}$ satisfy the self-consistent constraints as: $\mathbf{1}^{\mathrm{T}}\theta_l^{\mathbf{U}_b} = \theta_l^{v_b}\mathbf{1}^{\mathrm{T}}$ and $\sum_{k=1}^{K}\theta_l^{q_k} + \sum_{b\in\mathcal{B}_i}\theta_l^{v_b} = 1$, where $\theta_l^{v_b}$ is the summation result along each column of the matrix $\theta_l^{\mathbf{U}_b}$.

In summary, our system has two sets of parameters: the codebook $\{\mathbf{g}_l\}_{l=1}^{L}$ and $\Theta = \{\theta_l\}_{l=1}^{L}$, where each $\theta_l = \left\{\theta_l^{\mathbf{q}}, \left\{\theta_l^{\mathbf{U}_b}\right\}_{b\in\mathcal{B}}\right\}$ is the parameter associated with the basis $\mathbf{g}_l$ in the codebook. We next discuss codebook construction followed by parameter learning for Θ.

4. Construction of the Codebook $\{\mathbf{g}_l\}_{l=1}^{L}$

Our codebook construction is unsupervised. Given a set of image patch features $\{\mathbf{f}_i\}$ without label information, we compute the codebook $\{\mathbf{g}_l\}_{l=1}^{L}$ by minimizing the following error function.

$$\sum_i \sum_{l=1}^{L} c_{i,l}\|\mathbf{f}_i - \mathbf{g}_l\|^2 \quad (9)$$

We require that for each feature vector $\mathbf{f}_i$, its coefficient vector $\mathbf{c}_i = [c_{i,l}]_{l=1}^L$ has only R non-zero elements, each of value $\frac{1}{R}$. If $R = 1$, then Eq. (9) is a L-means clustering problem with a large number L. The result of L-means is L isolated clusters. Since we have a small number of labeled patches, many isolated clusters do not facilitate well label propagation for semi-supervised learning. Therefore, we require $R > 1$ to have a better-connected bipartite graph between nodes $\{\mathbf{f}_i\}$ and $\{\mathbf{g}_l\}$, as shown in Figure 3(a), where each non-zero $c_{i,l}$ defines an edge.

The algorithm for computing $\{\mathbf{g}_l\}_{l=1}^L$ is straightforward, which alternates between finding R nearest cluster centers for each $\mathbf{f}_i$ and updating each cluster center $\mathbf{g}_l$. To speed up the procedure for large data sets, we adopt an approximate R-NN algorithm using tree structure to partition the feature space. Specifically, we create R' trees, where $R' \geq R$; the leaf nodes of all the trees correspond to $\{\mathbf{g}_l\}$. Each tree r maps a feature vector $\mathbf{f}_i$ to a leaf node l_i^r. As a result, at each iteration, we only need to find R nearest cluster centers out of R' candidates (one from each tree query).

We find that trees whose node splitting criteria are based on a single feature component (*e.g.*, K-d trees or decision forests) are not effective for finding approximate nearest neighbors; instead we build vantage point trees (VP trees), which is one type of ball trees recommended in [10].

5. Learning Θ from Partially Labeled Images

Assuming that the codebook $\{\mathbf{g}_l\}_{l=1}^L$ has been constructed in advance without the need of label information, given a set of N training data $\{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^N$, where $\mathbf{x}_n$ is an image and $\mathbf{y}_n$ is a *partial* label map, we learn the parameter set Θ by finding an optimal Θ so that the "correctness" of computed label maps using Θ over labeled pixels is maximized. Mathematically, we minimize a regularized loss function:

$$\min_{\Theta} \sum_{n=1}^{N} \underbrace{\sum_{i \in \mathcal{L}_n} e_i(\Phi(\mathbf{x}_n;\Theta), \mathbf{y}_n)}_{E(\Phi(\mathbf{x}_n;\Theta), \mathbf{y}_n; \mathcal{L}_n)} + \lambda R(\Theta) \qquad (10)$$

where $\mathcal{L}_n$ is the set of *labeled* pixel indices in the training image $\mathbf{x}_n$, $e_i(\cdot,\cdot)$ is a loss function that compares the estimated label probability map $\Phi(\mathbf{x}_n;\Theta)$ and the corresponding training label map $\mathbf{y}_n$ at pixel i, and $R(\Theta)$ is a regularization function for parameter Θ with weight λ. Note that our formulation of loss function Eq. (10) requires only partially labeled images as training data.

Regularization $R(\Theta)$ The regularization function $R(\Theta)$ encourages similar bases $\mathbf{g}_{l_1}$ and $\mathbf{g}_{l_2}$ in the codebook to have similar parameters θ_{l_1} and θ_{l_2}, so that the label information in the training data can be propagated. We use the following heuristic to design the regularizer and leave other design choices as future work.

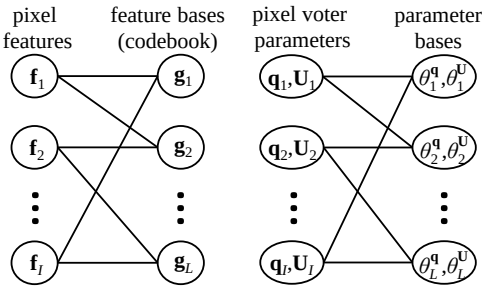


Figure 3: The sparse correspondence ($R = 2$) between patch features $\{\mathbf{f}_i\}$ and codebook bases $\{\mathbf{g}_l\}$ forms a bipartite graph in (a). The same graph structure connects pixel voter parameters and voter parameter bases in (b).

Specifically, if a feature vector $\mathbf{f}_i$ for pixel i is associated with R nearby cluster centers, we encourage these R cluster centers to have similar parameters by minimizing their parameter value variance. Recall in Eq. (7) and Eq. (8), the self voting vector $\mathbf{q}_i$ and label probability transition matrix $\mathbf{U}_{i,b}$ are defined as a weighted average of these parameters ($\theta_l^{\mathbf{q}}$ and $\theta_l^{\mathbf{U}_b}$) associated with the cluster centers. To minimize the variance, we define our regularizer as

$$R(\Theta) = \sum_{n=1}^{N} \sum_{i \in \mathcal{I}_n} \sum_{l=1}^{L} c_{i,l} \left(\|\mathbf{q}_i - \theta_l^{\mathbf{q}}\|^2 + \sum_{b \in \mathcal{B}} \|\mathbf{U}_{i,b} - \theta_l^{\mathbf{U}_b}\|^2 \right) \qquad (11)$$

Recall that for each i, only R components of $c_{i,l}$ are non-zero. As a result, the definition of $R(\Theta)$ in Eq. (11) propogates label information in the training data over the bipartite graph in Figure 3(b). Weighted averaging one group of nodes leads to optimal estimation of the other group of nodes to minimize $R(\Theta)$. For example, fixing $\{\theta_l^{\mathbf{q}}, \theta_l^{\mathbf{U}_b}\}$, weighted averaging based on Eq. (7) and Eq. (8) leads to optimal values of $\mathbf{q}_i$ and $\mathbf{U}_{i,b}$ that minimize $R(\Theta)$. The propagation effect strongly depends on the graph structure, *i.e.*, the non-zero weights of $[c_{i,l}]$. We evaluate the relationship between the propagation effect and the parameter settings in the experiment section.

Training Algorithm To minimize Eq. (10), we need to take its derivative with respect to Θ. $\frac{dE}{d\Phi}$ and $\frac{dR}{d\Theta}$ are straightforward to compute; $\frac{d\Phi}{d\Theta}$ is non-trivial to compute because $\Phi(\mathbf{x};\Theta)$ represents the stationary label distribution on a pixel network that uses Θ as parameters.

Although the derivative $\frac{d\Phi}{d\Theta}$ is inconvenient to evaluate analytically, numerical approximation can be computed efficiently. Specifically, in practice, we always use an iterative algorithm to compute the stationary distribution $\Phi(\mathbf{x};\Theta)$, such as Jacobi iteration in Eq. (5). Such algorithms can be

described as transforming the solution in a previous iteration, $\mathbf{y}^{(t-1)}$, to a new solution, $\mathbf{y}^{(t)}$:

$$\mathbf{y}^{(t)} = \Psi^{(t)}(\mathbf{y}^{(t-1)}; \mathbf{x}, \Theta) \tag{12}$$

where $\Psi^{(t)}$ is the transformation at iteration t that depends on the parameter Θ as well as the input $\mathbf{x}$. At each iteration, $\frac{d\mathbf{y}^{(t)}}{d\Theta}$ can be recursively evaluated using $\frac{d\mathbf{y}^{(t-1)}}{d\Theta}$ as

$$\frac{d\mathbf{y}^{(t)}}{d\Theta} = \frac{\partial \Psi^{(t)}}{\partial \mathbf{y}^{(t-1)}} \frac{d\mathbf{y}^{(t-1)}}{d\Theta} + \frac{\partial \Psi^{(t)}}{\partial \Theta} \tag{13}$$

where the partial derivatives $\frac{\partial \Psi^{(t)}}{\partial \mathbf{y}^{(t-1)}}$ and $\frac{\partial \Psi^{(t)}}{\partial \Theta}$ can be computed based on the chosen iteration scheme (*e.g.* Gauss-Siedel or Jacobi).

We set T to be the maximum number of iterations. We use the last iteration result $\mathbf{y}^{(T)}$ to approximate $\Phi(\mathbf{x}; \Theta)$ and use $\frac{d\mathbf{y}^{(T)}}{d\Theta}$ to approximate $\frac{d\Phi}{d\Theta}$. Eq. (13) defines a forward iteration of computing the Jacobian matrix $\frac{d\Phi}{d\Theta}$, which is slow because each iteration requires the matrix-matrix product $\frac{\partial \Psi^{(t)}}{\partial \mathbf{y}^{(t-1)}} \frac{d\mathbf{y}^{(t-1)}}{d\Theta}$ and both matrices become dense after many iterations. Note that we only need the product of $\frac{dE}{d\Phi}$ and $\frac{d\Phi}{d\Theta}$, which is a vector, instead of the Jacobian matrix $\frac{d\Phi}{d\Theta}$ itself. Inspired by the back propagation algorithm in neural net training, we can compute the product of $\frac{dE}{d\Phi}\frac{d\Phi}{d\Theta}$ efficiently using a backward iteration in which only vector-matrix products are required, as shown in Algorithm 1.[2]

Algorithm 1 Backward Iteration

Input: $g^{(T)} = \frac{dE}{d\Phi}$
Output: $a = g^{(T)} \frac{d\Phi}{d\Theta}$
  $a \leftarrow 0$
  for $t = T$ to 1 **do**
    $a \leftarrow a + g^{(t)} \frac{\partial \Psi^{(t)}}{\partial \Theta}\big|_{\mathbf{y}^{(t-1)}}$
    $g^{(t-1)} \leftarrow g^{(t)} \frac{\partial \Psi^{(t)}}{\partial \mathbf{y}^{(t-1)}}\big|_{\Theta}$
  end for
  $a \leftarrow a + g^{(0)} \frac{d\mathbf{y}^{(0)}}{d\Theta}$

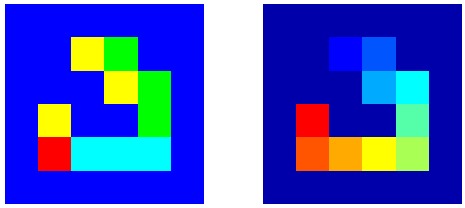
6. Experiments

We did three experiments to evaluate our voter model with missing labels, our regularization, and the overall method, respectively.

6.1. Experiments on Snake Dataset

The Snake dataset is used in [1, 15] to evaluate a context label interaction model with weak local evidence. A *snake* is a sequence of ten adjacent *snake pixels*, where the color of each snake pixel indicates the relative position of the next snake pixel in the sequence (red = *1 pixel above*, yellow = *1 pixel right*, cyan = *1 pixel left* and green = *1 pixel below*). The snake pixels are labeled 1 to 10 from head to tail, while the rest pixels (blue) are labeled 11 as background. A sample snake image is shown in Figure 4.

[2]The intuition behind backward iteration is: to compute $a \cdot B \cdot C$ where a is a vector (gradient) and B and C are matrices (Jacobian), it more efficient to compute $(a \cdot B) \cdot C$ than $a \cdot (B \cdot C)$.



(a) snake image (b) snake label

Figure 4: A sample snake image (left) and its corresponding ground truth label (right).

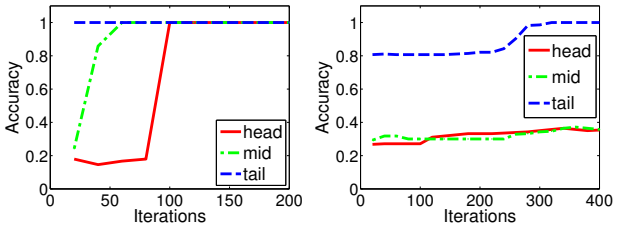
Table 1: The accuracy of snake head, mid, and tail estimation with different training image sizes and missing label rates.

missing label rate	training size = 20			training size = 60		
	Head	Mid	Tail	Head	Mid	Tail
0%	100%	100%	100%	100%	100%	100%
25%	100%	100%	100%	100%	100%	100%
50%	80.4%	100%	100%	100%	100%	100%
75%	35.4%	35.7%	100%	100%	100%	100%

It is clear that local features (pixel colors) have little correlation with the labels, *e.g.*, a red pixel can be any part of a snake given no context information. It is also worth noting that the difficulty of correctly labeling a snake pixel increases from tail to head, in particular with labels missing in the training data, because a correct head label depends on correct labels of all its successors, but a correct tail label does not depend on the labels of other snake pixels.

We test our method on the set of 300 images provided in [15], using both the first 20 images and the first 60 images as our training sets. For each of the two training sets, we control the missing label rates to be 0%, 25%, 50% and 75% at random pixel locations. The performance is reported in Table 1. Our method correctly labels all the snake pixels using only 20 fully labeled training images, which is as good as what is reported in [1], and requires much less training data (200) as used in [7] and [15]. Our method produces perfect results even when 25% of the labels in the 20 images are missing. The result starts to degenerate as we increase the missing rate to 50% and 75%. When we use the 60 images as training set, our method performs perfectly even when 75% of the labels are missing.

We are surprised by the convergence of our algorithm, especially in the presence of severe missing data. Figure 5 helps to explain the success of our method on this dataset. Even with a missing rate of 75%, so long as we have enough data, the algorithm can first label tail correctly at the beginning (< 20 iterations). Afterwards, mid pixels gradually



(a) training:60 missing:75% (b) training:20 missing:75%

Figure 5: The accuracy on testing data of intermediate model estimation in the training optimization process, sampled for every 20 iterations. In (a), with 60 training images, the label accuracy on tail pixels first converges to 1, and then propagate to mid pixels and head pixels. In (b), with the number of training images reduced to 20, the propagation is hindered and the label accuracy of mid pixels and head pixels converges at a relatively low level.

become correctly labeled, and tail pixels follow the trend.

6.2. Image Rotation Experiment

In this experiment we take an image from the CamVid dataset along with its ground truth label map. The image is then rotated 119 times with an interval of 3°, creating a series of 120 images. The four images with rotations of 0°, 90°, 180°and 270° are chosen as training images and are fully labeled. We ignore the context term $\mathbf{U}_{i,b}$ and only use the self vote term $\mathbf{q}_i$. Therefore, this experiment is a multi-class classification problem; we seek to illustrate the effect of parameter settings on codebook construction and label propagation using the regularization.

There are two independent variables in this controlled experiment: the depth of the vantage point forest, and the number of cluster centers R, with which each patch is associated. Increasing the depth of the vantage point forest expands the codebook, generating finer grained partition of the feature space. This increases the discriminative power of the codebook, but decreases the connectivity of the bipartite graph shown in Figure 3, and hinders the propagation. Decreasing R has a similar impact on the resulting bipartite graph. The trade-off between the quality of codebook bases and the connectivity of the bipartite graph indicates that there might exist an optimal combination.

In Figure 6 we compare the overall performance of different parameter settings. The baseline method directly uses the average label distribution [3] to vote for the associated patches. The result shows that increasing the depth of the forest as well as decreasing the clustering parameter boost the accuracy in general. However, for small clustering parameters, the performance suffers as tree depth increases, due to the poor connectivity of the bipartite graph.

[3]Unlabeled patches do not contribute to the average label distribution.

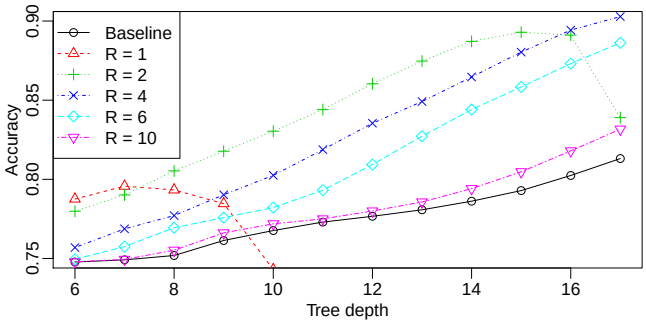


Figure 6: Performance comparison between our method with different parameter settings and the baseline method. It shows that with a fixed tree depth, the accuracy favors smaller clustering parameter R, when the tree is not too deep. However, a smaller R weakens connectivity, resulting in less satisfactory performance when the tree is deep.

6.3. Experiments on CamVid Dataset

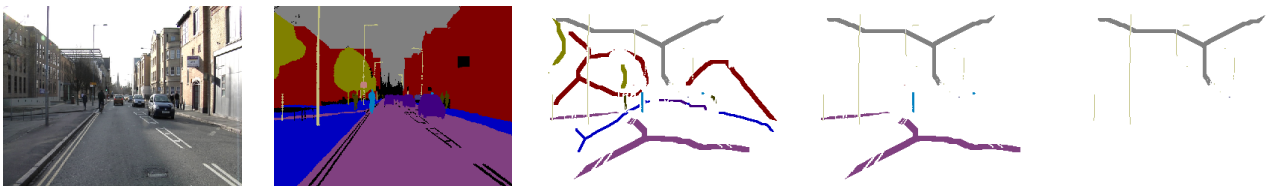
To demonstrate the performance of our model on stroke-labeled real-world images, we test our algorithm on the CamVid dataset [9]. This dataset consists of 600 annotated driving-scene images with 32 classes. Like many other literatures (*e.g.* [25, 1]), we choose only the most populated 11 classes and split all images into 367 training data and 233 testing data.

Our label strokes are generated from the training label maps using the following three steps:

1. Blurring each label map using a 21x21 large window to remove small isolated components for each semantic class region;
2. Applying a morphological thin operation to generate a skeleton of the semantic class regions;
3. Generating a stroke for each semantic class region by thickening (dilating) its skeleton until 10% of the pixels of the region are labeled.

To further mimic a realistic setting that a human annotator may not give strokes to all semantic classes in each image, we gradually and randomly discard some of the label strokes and use only k percent of the strokes in the training set. Examples of generated labels with different k are shown in Figure 7.

In this experiment, we extracted HOG features with a stride of 1. The cell size for HOG features is 9 x 9, and each block consists of 5 x 5 cells. We created 10 VP trees to construct the codebook for all 46 million patches from 600 CamVid images. Each tree is about 22 level deep, and contains roughly 0.5 million code words. We use the same baseline described in Section 6.2 (*i.e.*, the average label distribution without context interaction) to compare with our method. Note that strokes of all semantic classes are known in baseline (*i.e.*, $k = 100\%$). In the learning step, we adopt



(a) Original image (b) Groundtruth label (c) $k = 100\%$ (d) $k = 60\%$ (e) $k = 10\%$

Figure 7: Examples of generated label strokes with different percentages of known semantic classes. The original image is shown in (a), along with its groundtruth label in (b). This groundtruth label is processed with a blurring-thinning-dilating procedure, and forms our label strokes in (c). We further discard some label strokes until only k percent of strokes are labeled on average in the training set. Label strokes with $k = 60\%$ and $k = 10\%$ are shown in (d) and (e).

Table 2: Accuracy for most populated classes on CamVid. Baseline is the average label distribution without label probability transition matrix $\mathbf{U}_{i,b}(\mathbf{x}; \Theta)$ when $k = 100\%$.

	Sky	Road	Bldg	Sidewalk	Car	Tree
k=100%	88.1%	95.3%	86.8%	34.6%	28.2%	56.2%
k=60%	89.2%	95.5%	86.1%	33.7%	27.7%	55.8%
k=10%	88.6%	95.7%	80.9%	36.4%	21.2%	49.5%
baseline	83.0%	95.1%	83.8%	11.8%	4.3%	39.2%

hinge loss as loss function for Eq. (10), and use a 5 x 5 neighborhood. All the parameters are initialized as 1, followed by a normalization step to satisfy the self-consistent constraints in Eq. (4), and each element in the label probability vector is initialized as $\frac{1}{K}$.

A visual result can be viewed in our supplementary video. Table 2 shows the overall accuracy for most populated classes on CamVid dataset. Our overall accuracy is 75.0% for k=100% and 73.0% for k=10%. Compared with baseline, whose overall accuracy is 68.0%, our voter network model improves the accuracy on almost all populated classes. This improvement is attributed to the label interactions within neighborhood. In addition, as we gradually increase the labeled semantic classes on each image by increasing k, we find that additional strokes are mostly helpful for labeling "object classes" like cars and trees, but are less helpful for labeling "region classes" like sky and road. It suggests a more effective way of acquiring image labels is to ask users to draw strokes in object classes rather than region classes.

To further study the effectiveness of context interactions, we have designed another experiment. Given our probabilistic voter model, the context interaction is captured by the $K \times K$ non-negative matrix $\mathbf{U}_{i,b}(\mathbf{x}; \Theta)$. Therefore, by variating the size of the neighborhood $\mathcal{B}$ for each pixel, we can control how much context information is used to calculate the current label probability vector. We evaluate the overall accuracy of all the classes on the testing set and the result is reported in Table 3. It again shows that our voter network is effective in modeling neighborhood context.

Note also that our voter network method is relatively fast in training the parameters and inferring the labels for query

Table 3: Overall accuracy for different window size

window size	0 (baseline)	4	8	24
accuracy	68.0%	74.0%	74.8%	75.0%

image. On our test machine (dual Xeon E5-2670), the training takes about 30 minutes for every 20 iterations (gradient calculation and line search) using a cluster of 15 machines. In the testing stage, it takes around 1 second on average for each image using multi cores, with most of the time spent on feature generation and forest query.

7. Conclusion

In this paper, we present a novel voter network model to address the image parsing problem with only stroke labels. On small controlled synthetic experiments, we show that our framework is robust to handle severe missing labels in training data. On real experiments, we show that our framework is able to propagate label information in training data and to process a codebook of millions of bases.

We would also like to compare our Voter Network model with some other learning methods:

- Our method differs from existing energy-based structural learning, *e.g.*, structural SVM, in that our loss function is defined based on label probabilities of individual pixels while existing structural learning methods define loss using the energy value of an entire image. As a result, our method can easily handle training data with only partial labels. Furthermore, it naturally provides per-pixel label uncertainty (rather than per-image label map uncertainty).
- CRF is another common tool used for semantic segmentation. Learning the parameters of a CRF can be tricky because of the normalization factor. CRF inference can be challenging as well. Our parameter learning does not involve the normalization factor and our inference (based on Gaussi-Seidel or Jacobi iteration) is straightforward and simple.
- The backward iteration (Algorithm 1) of derivative calculation in our paper is similar to the Back Propagation algorithm in neural nets because both are applications

Table 4: Comparison between our method and alternating approach on the snake dataset. The alternating approach fails even when missing label rate is relatively low, *e.g.* at 10%. Our method generates correct estimation even when the missing rate reaches 25%.

	Head	Mid	Tail
Alternating (missing = 1%)	84.29%	95.36%	99.64%
Alternating (missing = 5%)	91.07%	95.36%	99.29%
Alternating (missing = 10%)	22.50%	26.79%	44.29%
Alternating (missing = 25%)	9.64%	12.86%	27.50%
Ours (missing = 25%)	100%	100%	100%

of the chain rule for computing derivatives. In fact, each iteration in the Jacobi (or Gauss-Seidel) iteration is analogous to one layer of a neural net as it maps one solution state to another solution state. However, our approach is fundamentally different from neural nets in two ways. First, our iteration steps correspond to a chosen optimization algorithm in a principled way, while neural net architectures (number of layers, neuron connectivity, response function) are designed *ad hoc*. Second, every iteration step in our approach shares the same set of parameters to be estimated while each layer in a neural net has its own parameters, which makes parameter learning of a neural net prone to local minima.

- A common approach to handle partially-labeled data is alternating between updating missing labels and model parameters. This approach is susceptible to local minima. However, our optimization algorithm simultaneously estimates model parameters and missing labels, making it less sensitive to local minima. A comparison is reported in Table 4.

There are several interesting future research topics, including (1) generalizing a neighborhood in a pyramid to take into account context in scale space, (2) increasing the system scalability to handle millions of images, (3) handling thousands of class labels, and (4) exploring label uncertainty in interactive labeling of large image collection (crowd sourcing).

References

[1] S. Bulo, P. Kontschieder, M. Pelillo, and H. Bischof. Structured local predictors for image labeling. In *CVPR*, 2012. 2, 5, 6

[2] P. Clifford and A. Sudbury. A model for spatial conflict. *Biometrika*, 60(3):581 – 588, 1973. 1

[3] C. Galleguillos, A. Rabinovich, and S. Belongie. Object categorization using co-occurrence, location and appearance. In *CVPR*, 2008. 2

[4] J. Gonfaus, X. Boix, J. van de Weijer, A. Bagdanov, J. Serrat, and J. Gonz andlez. Harmony potentials for joint classification and segmentation. In *CVPR*, 2010. 2

[5] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *CVPR*, 2004. 2

[6] A. Jain, L. Zappella, P. McClure, and R. Vidal. Visual dictionary learning for joint object categorization and segmentation. In *ECCV*, 2012. 2

[7] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression tree fields - an efficient, non-parametric approach to image labeling problems. In *CVPR*, 2012. 2, 5

[8] P. Kohli, L. Ladicky, and P. Torr. Robust higher order potentials for enforcing label consistency. In *CVPR*, 2008. 2

[9] P. Kontschieder, S. Bulo, H. Bischof, and M. Pelillo. Structured class-labels in random forests for semantic image labelling. In *ICCV*, 2011. 2, 6

[10] N. Kumar, L. Zhang, and S. K. Nayar. What is a Good Nearest Neighbors Algorithm for Finding Similar Patches in Images? In *ECCV*, 2008. 4

[11] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. Graph cut based inference with co-occurrence statistics. In *ECCV*, 2010. 2

[12] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. In *SIGGRAPH*, 2004. 1

[13] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *TPAMI*, 33(12), 2011. 2

[14] F. Moosmann, B. Triggs, and F. Jurie. Scene segmentation with conditional random fields learned from partially labeled images. In *NIPS*, 2007. 2

[15] S. Nowozin, C. Rother, S. Bagon, T. Sharp, B. Yao, and P. Kohli. Decision tree fields. In *ICCV*, 2011. 2, 5

[16] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004. 1

[17] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008. 1

[18] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, 2006. 2

[19] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *CVPR Workshop*, 2008. 1

[20] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *ECCV*, 2010. 2

[21] J. Verbeek and B. Triggs. Region classification with markov field aspect models. In *CVPR*, 2007. 2

[22] A. Vezhnevets and J. Buhmann. Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. In *CVPR*, 2010. 2

[23] A. Vezhnevets, V. Ferrari, and J. Buhmann. Weakly supervised semantic segmentation with a multi image model. In *ICCV*, 2011. 2

[24] A. Vezhnevets, V. Ferrari, and J. Buhmann. Weakly supervised structured output learning for semantic segmentation. In *CVPR*, 2012. 2

[25] Y. Yang, Z. Li, L. Zhang, C. Murphy, J. V. Hoeve, and H. Jiang. Local label descriptor for example based semantic image labeling. In *ECCV*, 2012. 2, 6