Machine Learning of Image Analysis with Convolutional Networks and Topological Constraints

by

Viren Jain

B.A.S., Computer Science, University of Pennsylvania (2004)
B.A., Cognitive Science, University of Pennsylvania (2004)

Submitted to the Department of Brain & Cognitive Sciences
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010

Author . . . . . .
Department of Brain & Cognitive Sciences
December 10, 2009

Certified by . . . . . .
H. Sebastian Seung
Professor of Computational Neuroscience
Thesis Supervisor

Accepted by . . . . . .
Earl K. Milller
Chairman, Department Committee on Graduate Theses

Machine Learning of Image Analysis with Convolutional Networks and Topological Constraints

by

Viren Jain

Submitted to the Department of Brain & Cognitive Sciences
on December 10, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computation

Abstract

We present an approach to solving computer vision problems in which the goal is to produce a high-dimensional, pixel-based interpretation of some aspect of the underlying structure of an image. Such tasks have traditionally been categorized as "low-level vision" problems, and examples include image denoising, boundary detection, and motion estimation. Our approach is characterized by two main elements, both of which represent a departure from previous work. The first is a focus on convolutional networks, a machine learning strategy that operates directly on an input image with no use of hand-designed features and employs many thousands of free parameters that are learned from data. Previous work in low-level vision has been largely focused on completely hand-designed algorithms or learning methods with a hand-designed feature space. We demonstrate that a learning approach with high model complexity, but zero prior knowledge about any specific image domain, can outperform existing techniques even in the challenging area of natural image processing. We also present results that establish how convolutional networks are closely related to Markov random fields (MRFs), a popular probabilistic approach to image analysis, but can in practice can achieve significantly greater model complexity.

The second aspect of our approach is the use of domain specific cost functions and learning algorithms that reflect the structured nature of certain prediction problems in image analysis. In particular, we show how concepts from digital topology can be used in the context of boundary detection to both evaluate and optimize the high-order property of topological accuracy. We demonstrate that these techniques can significantly improve the machine learning approach and outperform state of the art boundary detection and segmentation methods.

Throughout our work we maintain a special interest and focus on application of our methods to connectomics, an emerging scientific discipline that seeks high-throughput methods for recovering neural connectivity data from brains. This application requires solving low-level image analysis problems on a tera-voxel or peta-voxel scale, and therefore represents an extremely challenging and exciting arena for the development of computer vision methods.

Thesis Supervisor: H. Sebastian Seung
Title: Professor of Computational Neuroscience

Contents

List of Figures

Chapter 1

Introduction

The field of computer vision engages the problem of developing algorithmic solutions to the interpretation of data acquired from an optical source. This definition is necessarily general as the field encompasses a diverse array of specific problems and domains. For example, the optical source could be:

- a digital camera
- a medical imaging device (e.g., MRI or CT)
- microscopy (e.g., an electron microscope)

while the resulting "interpretation" may include:

- classification of an object in the input image (e.g., optical character recognition)
- reconstruction of the physical structure of an object within an image
- a decision concerning navigation (e.g., "turn left to avoid the tree").

There has traditionally been a distinction between "low-level" and "high-level" computer vision problems. Examples of high-level tasks include object recognition and visually guided robotic navigation. A common theme in high-level visual tasks is the conversion of a high-dimensional input image into a low-dimensional decision or classification. In contrast, low-level visual tasks are those in which the output is also a high-dimensional object that repre-

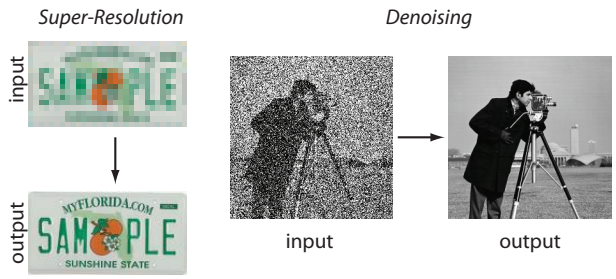


Figure 1-1: Example of two classic low-level vision computations: super-resolution and denoising.

sents some kind of interpretation of the input image. Examples of such tasks include image denoising, super-resolution, boundary detection, and motion estimation.

The practical importance of low-level computer vision is not difficult to establish. Image enhancement and segmentation techniques could potentially save lives by making the interpretation of medical imaging more efficient and accurate. Super-resolution techniques are of consistent interest to state security and law enforcement agencies due to the abundance of low-resolution images resulting from satellites or cell-phones that could be enhanced to provide crucial information. We will shortly discuss how boundary detection techniques are poised to make fundamental contributions to the reverse engineering of brains.

The full extent of the practical scenarios just mentioned have not been realized due to the lack of low-level vision techniques with the appropriate characteristics. In particular, it has proven difficult to develop highly *accurate* low-level vision algorithms. Traditionally in computer science, the accuracy of a computation is ensured by defining mathematical criteria for correctness and then proving that a proposed algorithm produces results consistent with the criteria. In low-level vision, this is not usually possible because the computational problems are fundamentally ill-posed, in the sense that the solution is not unique and may not depend continuously on the data [6]. Therefore, it is necessary to exploit regularization techniques and prior knowledge to solve a low-level vision problem [81, 48].

In many previous approaches to low-level vision, prior knowledge is hypothesized based on studying the structure of images and then manually encoded into the behavior of an algorithm [19, 31]. A well-known example is the Canny edge detector, which was derived by formulating several criteria for a good edge detector given simple assumptions about image

noise and the structure of edges (defined as an abrupt change in some low-level image feature) [19]. This approach, while influential and surprisingly effective given its simplicity, has found limited utility in real-world vision applications due to the difficulty in forming a higher-level interpretation of visual scene structure from the results of an unreliable edge detector. In particular, the true goal may not be to detect an edge, but rather a *boundary*, which is defined as a contour that separates one object from another [70]. The way boundaries appear in real images resist a simple mathematical specification. Edges may be one cue useful in the detection of a boundary, but are neither necessary nor sufficient.

1.1 A Machine Learning Approach to Low-Level Vision

A recent trend in low-level vision problems has been to use machine learning techniques that adapt the parameters of an algorithm to a particular dataset and problem. The basic philosophy of the machine learning approach is that instead of specifying exactly what to compute and how to compute it, humans provide *examples* of the desired computation. These examples are used to guide 'learning,' a process in which a parametrized function is optimized using a set of training examples and a cost function which quantifies performance. The hope is that by learning how to correctly interpret the training examples, the optimization will produce a function which can successfully generalize to other examples of the task not found in the training set. The creation of ground-truth databases has been a major factor in the growth of data-driven approaches to low-level vision [68]. Without such databases, it is difficult to quantify the performance of a method, much less optimize it using learning.

A major advantage of the machine learning approach to image analysis is *adaptability*. Computer vision methods are today applied to a wide variety of situations in which an image may have been formed using photons, electrons, magnetic resonance, or other means. The image itself may depict nanoscale cellular structure, macroscopic organ structure, the natural world, or other phenomena. This diversity presents a challenge for the computer vision community, because algorithms developed with one imaging context in mind may not be effective in another context. For example, several decades of research into the statistics of natural images have produced sufficient understanding to design natural image denoising

algorithms that work remarkably well [82]. However, the assumptions behind this algorithm, gleaned from careful study of natural images, are unlikely to be true in more specialized imaging situations where one might also be interested in denoising.

Natural images have been well studied due to their obvious importance and abundance, as well as interest in the human visual system. For many other datasets (such as those arising from microscopy), there is little interest in understanding the detailed low-level statistics of the imagery. Therefore, it is desirable to have computer vision methods which can perform well in the absence of a sophisticated understanding of the underlying structure of the images.

Machine learning presents an alternative approach which addresses the issue of domain specificity. Instead of hand-embedding prior knowledge into an algorithm, learning can adapt a general algorithm to the specifics of a dataset. This approach requires creation of a training set with examples of the desired transformation. For example, in the electron microscopy restoration problem studied in Chapter 5, the desired 'clean images' are created by human interpretation, as we only have access to the noisy input images. Conversely, in the natural image denoising studied in Chapter 4 it is the *noisy* images which are generated (by applying a noise model to a database of clean images).

1.1.1 Convolutional Networks

The first major claim of this thesis is that machine learning can be used to solve low-level vision problems with greater accuracy than techniques based on hand-encoded prior knowledge if an appropriately flexible classifier architecture is used. In Chapter 2 of this thesis, we develop convolutional networks as a flexible classifier architecture capable of achieving good performance, and in Chapters 4 and 5 we show that less powerful classifiers do not achieve as good performance. We demonstrate that convolutional networks, which are not specialized for a specific image domain, can meet or exceed the performance of techniques that were specifically tailored to some domain.

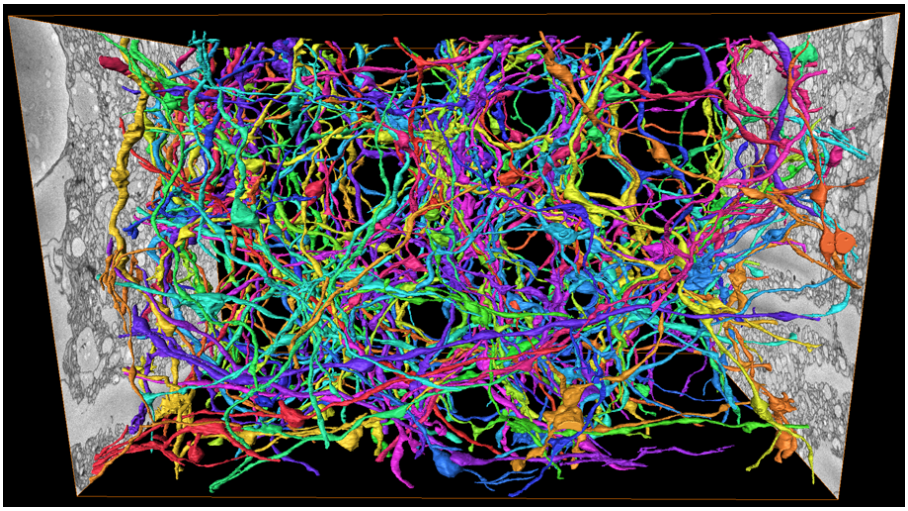


Figure 1-2: Putative neurites automatically traced using techniques described in this thesis, in a volume of rabbit retina imaged with Serial Block Face Scanning Electron Microscopy (SBF-SEM) [24, 18].

1.1.2 Topological Learning

The second major claim of this thesis is that the machine learning of a specific low-level problems can be greatly improved by the use of cost functions and learning algorithms that reflect the structure of the desired interpretation space. We study the specific problem of boundary detection, and develop a learning algorithm that optimizes the high-order property of topological accuracy. We build on concepts from digital topology to introduce the warping error (Chapter 7) and its associated optimization algorithm (Chapter 8) as a means of accomplishing this.

1.2 Connectomics: Reverse Engineering the Brain

A major motivation for the development of the methods in this thesis has been provided by the scientific ambition of reconstructing wiring diagrams of biological nervous systems. In this section we describe this agenda and how computer vision research plays a fundamental role in its pursuit.

A significant assumption behind many models and theories of neural computation is that the function of a neural circuit is closely related to its structure at the level of synaptic

connections. In order to test both this general assumption and the many specific structure-function relationships that have been hypothesized, it would be ideal to be able to study connectivity patterns within the brain. This entails the collection of *connectomes*, which are complete maps of all the connections in a brain or a piece of brain [61, 99, 97]. In order to recover a connectome from a piece of tissue, two fundamental tasks must be performed:

1. The tissue must be imaged at a sufficient resolution to be able to see axons, dendrites, and synapses, and over a sufficient field of view to contain the circuitry of a functionally significant assembly of cells.

2. The resulting images must be analyzed so that it is known which cell each axon and dendrite belongs to, and synapses must be identified to establish a connection between two cells.

Unfortunately, technical challenges involved in both these steps have thus far prevented the collection of connectomes. An exception is the connectome of the small worm *C. elegans*. The 7000 connections of the 300 neurons of the *C. elegans* brain were mapped using electron microscopy imaging and human annotation over a period of ten years in the 1970's, with some additional recent effort to finalize and complete the map [111, 21, 107]. Scaling this approach to organisms with hundreds of thousands or millions of neurons is clearly impractical (which is to say nothing of a human connectome, which contains on the order of a hundred billion neurons and trillions of connections).

The past several years have seen a major effort to overcome technical barriers associated with the collection of the raw imaging data necessary for producing connectomes. Various efforts to improve and automate serial section electron microscopy (EM) have yielded new methods which are likely to make the collection of the raw images necessary for reconstructing wiring diagrams a significantly more successful and attractive proposition [24, 18, 37, 49, 40]. Other major developments include techniques for nanometer-scale imaging using light microscopy [93, 39], multicolor fluorescent labeling techniques that provide some amount of information on neurite identity directly from an image [62], and fixation methods for repeated antibody staining and simultaneous electron and light microscopy [72].

After an image volume has been collected, then the hard work of analysis begins. In

particular, tracing each individual axon and dendrite to its parent cell body, throughout an image volume that may consist of billions or trillions of voxels, is a painstaking task. Although it is possible to trace the wires manually, this approach involves prohibitive amounts of human labor (the majority of the ten years of effort for *C. elegans* were consumed by the tracing task). Since existing techniques are insufficiently accurate to solve this problem, the automated analysis of EM images of brain tissue represents an intriguing opportunity for advances in computer vision.

The challenging computer vision problems posed by the connectomics agenda have been a driving force for the development of novel image processing strategies, both in this thesis and in other recent work [103, 2, 46, 73, 12, 65]. However, the techniques introduced in this thesis are fundamentally general and applicable to a wide variety of imaging domains.

1.3 Summary of Contributions

In this thesis, we develop a novel machine-learning approach to low-level computer vision that is applicable to problems such as image denoising, restoration, and segmentation. Several specific contributions are made:

- Convolutional networks are developed as a computational architecture for low-level image processing (Chapter 2).
- Mathematical connections between convolutional networks and Markov Random Fields are exposed (Chapter 3).
- Convolutional networks with zero domain-specific prior knowledge, other than translation invariance, are used to achieve state of the art performance on the tasks of natural image denoising and restoration of electron microscopy images (Chapters 4 and 5).
- Concepts from digital topology are used to introduce "warping error:" a novel metric for evaluating segmentation performance (Chapters 6 and 7).
- Warping error is optimized with supervised learning to yield superior boundary detection and image segmentation performance, as compared to a a variety of state of the

art methods (Chapter 8).

Finally we note that parts of this thesis are joint work with others and have appeared in [44, 45, 12].

Chapter 2

Convolutional Networks

Abstract. We provide a detailed description of convolutional networks, the machine learning architecture used extensively in this thesis. Convolutional network pursue the machine learning approach by making nearly all details of the algorithm subject to optimization during learning. In fact, convolutional networks exploit learning more than many other approaches. A traditional pattern recognition strategy is to hand-specify specific features to compute from the input signal, and then use learning to optimize the manner in which features are combined into a prediction. This approach has recently been applied to boundary detection as well [25, 71, 70]. As explored in this thesis, however, convolutional networks are applied to the image directly. Therefore, the learning procedure discovers both what feature space to use *and* how to use it. In Chapters 4 and 5 we empirically compare the accuracy of this approach to other methods that rely less on learning and more on hand-encoded prior knowledge.

2.1 Comparison to previous work

The present work is related to the extensive literature on applications of neural networks to image processing, which has been reviewed by Egmont-Petersen et al [27]. In this literature, a multilayer perceptron is applied to patches of the input image. Our work is distinct from this literature because our networks are convolutional.

We were inspired by previous research on convolutional networks applied to object recog-

nition, as well as one previous study that used convolutional networks to label and segment regions in microscopic imagery [56, 58, 75]. In addition to convolutions, all of these networks also included subsampling, which produced an output representation of much lower resolution than the input image. Subsampling is an important strategy in object recognition, where it helps achieve invariance to distortions of the visual image by discarding positional information about image features and details. But many image processing applications *require* precise positional information. The segmentation of fine branches of neurons in EM images is a good example. Therefore our convolutional networks do not include subsampling, and we expect that this will be appropriate for many other image processing applications. Indeed, for the application to EM images, we have introduced the use of *supersampling* in our networks to increase the resolution of the output image. This is critical because the spacing between objects sometimes narrows to less than one pixel of the input image.

2.2 Network Architecture and Dynamics

A convolutional network is an alternating sequence of linear filtering and nonlinear transformation operations. The input and output layers include one or more images, while intermediate layers (referred to as "hidden" layers) contain images called feature maps that are the internal computations of the algorithm. The activity of feature map a in layer k is given by

$$I_a^k = f(u_a^k) \tag{2.1}$$

$$u_a^k = (\sum_b w_{a,b}^k \otimes I_b^{k-1}) + b_a^k \tag{2.2}$$

where I_b^{k-1} are feature maps that provide input to I_a^k, and $\otimes$ denotes the convolution operation. The function f is some differentiable nonlinearity; typically we use the logistic function $f(x) = 1/\left(1 + e^{-x}\right)$. b_a^k is a bias parameter. Figure 4-1 depicts a convolutional network with four hidden layers, each with 24 feature maps. Each arrow in the network represent a single convolution and is therefore associated with a single filter (in this case, a 5×5 2d filter).

Feature maps in a convolutional network are typically organized into layers, such that

computation of the output image depends on computing the activities in the last hidden layer, which depends on compute the activities in the penultimate hidden layer, and so on. Inference (the process of computing the output of the model for a novel input) requires computing the activity of the output layer given some input, which in neural network terminology means computing the "forward pass" of the network.

In Chapter 5 we demonstrate how this simple architecture can be modified to perform super-resolution image processing by having feature maps organized into a super-sampling operation.

2.3 Learning

2.3.1 Supervised learning with gradient descent

In this thesis we employ supervised learning, which requires four components:

1. Learning machine: a function $F_{\vec{\theta}}$ with a vector of free parameters $\vec{\theta}$.
2. Training set $\{x, y\}_i$ with inputs x and ground truth y.
3. Scalar-valued cost function $E(x, y, F_{\vec{\theta}})$.
4. Learning algorithm that minimizes E with respect to $\vec{\theta}$.

We have already introduced convolutional networks as our chosen function F, and will discuss training sets in more detail in Chapters 4 and 5. In this section we focus on how to optimize convolutional networks in terms of a cost function using a learning algorithm.

The convolutional network architecture is highly non-linear and non-convex, which rules out convex optimization methods. However, it was designed to be fully differentiable with respect to each of the free parameters of the network, which leaves open the possibility of gradient optimization as long as we choose a cost function that is similarly differentiable. We will typically use the sum-of-squared errors function (or some minor variant of it):

$$E(x, y, F_{\vec{\theta}}) = \sum_{i=1}^{N} (y_i - F_{\vec{\theta}}(x_i))^2$$

where i indexes over the N training examples. We are therefore interested in performing the following optimization:

$$argmin_{\vec{\theta}} E(x, y, F_{\vec{\theta}})$$

for which a local minimum can be found using gradient descent, a first-order optimization algorithm. Assuming that $\nabla_{\vec{\theta}} E$ can be calculated, the following general update can be iterated to minimize E:

$$\vec{\theta} \leftarrow \vec{\theta} - \eta \nabla_{\vec{\theta}} E(x, y, F_{\vec{\theta}}) \tag{2.3}$$

for some sufficiently small value of η.

2.3.2 Backward Pass

Having established the basic outline of gradient optimization, we derive the specific form of $\nabla_{\vec{\theta}} E_{\vec{\theta}}(x, y, F_{\vec{\theta}})$ in terms of the convolutional network architecture specified in Section 2.2. We will express $\nabla_{\vec{\theta}} E_{\vec{\theta}}(x, y, F_{\vec{\theta}})$ in terms of the individual components of $\vec{\theta}$ in convolutional networks: $\{w_{k,ab}, b_{k,a}\}$ for all k, a, b. For convenience, we abbreviate $E(x, y, F_{\vec{\theta}})$ with E in the following. Then from Eq. 2.1 it is clear that:

$$\frac{\partial E}{\partial w^k_{a,b}} = \sum_{i=1}^{N} \frac{\partial E}{\partial (u^k_a)_i} \frac{\partial (u^k_a)_i}{\partial w^k_{a,b}} \tag{2.4}$$

where i indexes over all spatial locations, and the sum is because all spatial locations within the same feature map share the same weights. However, we know from Equation 2.1 that

$$\frac{\partial u^k_a}{\partial w^k_{a,b}} = I^{k-1}_b.$$

Thus our primary task is to compute $\frac{\partial E}{\partial (u^k_a)_i}$, a quantity so important that we give it a special designation: the "sensitivity." It will also be convenient to define the "sensitivity map", or S^k_a, which is the collection of all sensitivities that corresponds to feature map a in layer k. It is instructive to consider a few things about the sensitivity map:

1. S^k_a is an "image" with the same size and dimension as the activity feature map I^k_a.

2. $S_a^k = \frac{\partial E}{\partial u_a^k} = \frac{\partial E}{\partial u_a^k}\frac{\partial u_a^k}{\partial b_a^k} = \frac{\partial E}{\partial b_a^k}$ which is due to the fact that $\frac{\partial u_a^k}{\partial b_a^k} = 1$.

Thus the sensitivity map immediately gives us the gradient of the cost function with respect to the biases of any unit in the feature map.

The sensitivity map can be computed in a matter analogous to the activities except in the reverse order, which is why in neural network terminology this computation is usually called the "backward" pass. If a network has L layers with $L-1$ hidden layers, then the sensitivity S^L of the output layer $I^L = F_\theta(x_i)$ is given by

$$S_a^L = \frac{\partial E}{\partial u_a^L} = \frac{\partial \frac{1}{2}\sum_i (y_a - I_a^L)_i^2}{\partial u_a^L} = (y_a - I_a^L)\frac{\partial I_a^L}{\partial u_a^L} = (y_a - I_a^L) \odot f'(u_a^L)$$

where $\odot$ denotes pixel-wise multiplication, and we have assumed that the final output layer may have multiple feature maps, each with its own target y_a in the training set. In many situations, however, there is only a single output feature map. We would now to like to derive an expression for the sensitivities in the second to last layer:

$$S_a^{L-1} = \frac{\partial E}{\partial u_a^{L-1}} = \sum_k (\frac{\partial E}{\partial u_k^L}\frac{\partial u_k^L}{\partial u_a^{L-1}}) = \sum_k (S_k^L \circledast w_{k,a}^L) \odot f'(u_a^{L-1}) \tag{2.5}$$

where $\circledast$ denotes cross-correlation and we have thus obtained a general recursive relationship for S_a^{L-1} in terms of S^L.

Detailed one-dimensional case

The appearance of this last expression in Equation 2.5 may seem a bit puzzling at first. To better understand it, we consider in detail the case of a 1-D convolutional network:

$$(u_a^l)_i = (w_{a,p}^l \otimes I_p^{l-1})_i$$

where $(u_a^L)_i$ is the preactivation of the i'th spatial location in feature map a in layer l, and we have defined its activity to depend on just a single feature map p in layer $l-1$. By the

definition of (one-dimensional) convolution:

$$(w^l_{a,p} \otimes I^{l-1}_p)_i = \sum_{x=-\infty}^{\infty} (I^{l-1}_p)_{i-x}(w^l_{a,p})_x$$

where for now we will ignore issues of zero-padding, etc. Suppose now we would like to compute the sensitivity S^{l-1}_p. Recall that:

$$S^{l-1}_p = \frac{\partial E}{\partial u^{l-1}_p} = \sum_k (\frac{\partial E}{\partial u^l_k}\frac{\partial u^L_k}{\partial u^{L-1}_p}) = \frac{\partial E}{\partial u^l_a}\frac{\partial u^L_a}{\partial u^{l-1}_p}$$

where we eliminate the sum as we assume only a single feature map in both layers l and $l-1$. Then for a single spatial location j in S^{l-1}_p:

$$(S^{l-1}_p)_j = \sum_{y=-\infty}^{\infty} \frac{\partial E}{\partial (u^l_a)_{j+y}}\frac{\partial (u^l_a)_{j+y}}{\partial (u^{l-1}_p)_j}$$

where by definition $\frac{\partial E}{\partial (u^l_a)_{j+y}} = (S^l_a)_{j+y}$ and

$$\frac{\partial (u^l_a)_{j+y}}{\partial (u^{l-1}_p)_j} = \frac{\partial (\sum_{x=-\infty}^{\infty} (I^{l-1}_p)_{j+y-x}(w^l_{a,p})_x)}{\partial (u^{l-1}_p)_j} = \frac{\partial (I^{l-1}_p)_j (w^l_{a,p})_y}{\partial (u^{l-1}_p)_j} = f'(u^{l-1}_p)_j (w^l_{a,p})_y$$

thus:

$$(S^{l-1}_p)_j = f'(u^{l-1}_p)_j \sum_{y=-\infty}^{\infty} (S^l_a)_{j+y}(w^l_{a,p})_y = (S^l_a \circledast w^l_{a,p})_j f'(u^{l-1}_p)_j$$

by the definition of cross-correlation. And thus the full sensitivity map is given by:

$$S^{l-1}_p = (S^l_a \circledast w^l_{a,p}) \odot f'(u^{l-1}_p).$$

In higher dimensions, the indexing becomes more complicated, but can still be reduced to multi-dimensional cross-correlation.

2.3.3 Update pass

Having computing sensitivities via the backward pass, updates can be made to the weights and biases:

$$\frac{\partial E}{\partial w^k_{a,b}} = I^{k-1}_b \circledast S^k_a \tag{2.6}$$

$$\frac{\partial E}{\partial b^k_a} = \sum_i (S^k_a)_i \tag{2.7}$$

where i indexes over all spatial locations in the sensitivity map. These gradients can be used in Equation 2.3 to update the parameters of the network.

Detailed one-dimensional case

To understand the origin of the cross correlation in equation 2.6, we can again consider the one-dimensional case. Recall that within a feature map, all spatial locations i share the same w; thus when computing the gradient for any particular w all of the gradients at separate spatial locations must be summed together. A *single* location j in filter $w^l_{a,p}$ will be denoted as $(w^l_{a,p})^j$, and the gradient of this parameter is given by

$$\frac{\partial E}{\partial (w^l_{a,p})^j} = \sum_{x=-\infty}^{\infty} \frac{\partial E}{\partial (u^l_a)_x} \frac{\partial (u^l_a)_x}{\partial (w^l_{a,p})^j} = \sum_{x=-\infty}^{\infty} (S^l_a)_x (I^{l-1}_p)_{j+x} = (I^{l-1}_p \circledast S^l_a)_j$$

and therefore the entire filter is given by

$$\frac{\partial E}{\partial w^l_{a,p}} = I^{l-1}_p \circledast S^l_a$$

where an identical relation holds true in higher dimensions, except using multidimensional cross-correlation.

2.3.4 Batch versus online learning

As defined in Equation 2.3, gradient-based supervised learning involves computing the gradient over *all* training examples, combining the gradients by averaging, and then making a single update. This process is then repeated many times until convergence. From the mathematical point of view this approach (sometimes refereed to as *batch* learning) is clearly justified and guaranteed to converge to a local minimum.

However, in practice it is often inefficient to compute the gradient for all the training examples just to make a single update. An alternative is *online* learning, in which the gradient for a single example is computed in order to make an update. Although any single update may be very noisy (with respect to the true gradient), the convergence guarantees stem from the fact that on average the update will point in the direction of the gradient [57, 13]. In practice, this approach has often been found to be significantly more efficient for training large neural networks which may require many hundreds of thousands of gradient updates to near convergence [56, 58].

In this work, we generally exploit a method known as *minibatch* learning, in which a small subset of examples from the training set are used to approximate the gradient (instead of just a single example as in online learning). Despite the name, the method is therefore more similar to online methods than true batch learning. As with online learning, minibatch learning is guaranteed to converge (without bias though with possibly high variance) if the particular set of examples are chosen i.i.d. from the training set.

Despite the theoretical requirement of i.i.d. samples, the most efficient (in the sense of amount of computation required to reach a given training error) procedure we found was one in which we chose a spatially localized *patch* of examples. Because our training sets typically represent image-to-image transformations (such as a noisy image to a clean image), a single image can generate many training examples due to the various spatial locations. We combined several spatially localized examples within the training set to form an update (for example, a $6 \times 6 \times 6$ cube of examples for 3d transformations, or a 6×6 patch of examples for 2d transformations). This approach was an especially efficient way of assembling a minibatch due to the fact that neighboring pixels share computations in a convolutional network.

2.3.5 Rebalancing training sets

In the binary classification task studied in Chapter 5, the dataset was unbalanced in the sense that there were roughly 80% positive examples and 20% negative examples. To obtain the best performance, it was necessary to *rebalance* the training set by sampling the negative examples more highly than the positive examples. By "best" performance, we mean both the lowest training error and lowest generalization error. In other words, optimizing on the

rebalanced training set led to a lower minimum error on the *unbalanced* training set, which is an interesting local minima phenomenon that has been previously reported in the literature [57].

2.3.6 Learning rates

Efficient convergence to a low training error local minimum in neural network learning is dependent on the appropriate choice of learning rates. Selecting learning rates is often a process of trial and error; we explored more sophisticated methods such as using an online approximation to the diagonal of the Hessian (discussed in [57]), but observed no benefit.

For binary image restoration, we found good results using the following procedure: η was set to 0.1 for all layers and we normalized the gradient in 2.4 by N, the number of units that share the same weights.

For natural image denoising, in which the targets are real-valued numbers, we neglected the normalization by N and used an η of 0.1. However, for networks with more than one hidden layer we used an η of 0.001 in just the last layer (the output layer). We speculate that smaller learning rates are required due to the precise analog targets in image denoising.

2.4 Filters, Layers, Field of View, and Context

In a low-level task such as boundary detection, local ambiguity can require the use of non-local context to form an unambiguous interpretation. In severe cases, truly global information may be required to resolve a local ambiguity. Appropriate use of context is thus a fundamental issue in designing an image processing strategy, and poses several challenges:

1. Determining the amount of context required to achieve high accuracy in the desired task.

2. Devising a computationally efficient way to exploit the required amount of image context.

3. In machine learning approaches, sample complexity: the amount of training data required to generalize accurately may depend on both the amount of context used and

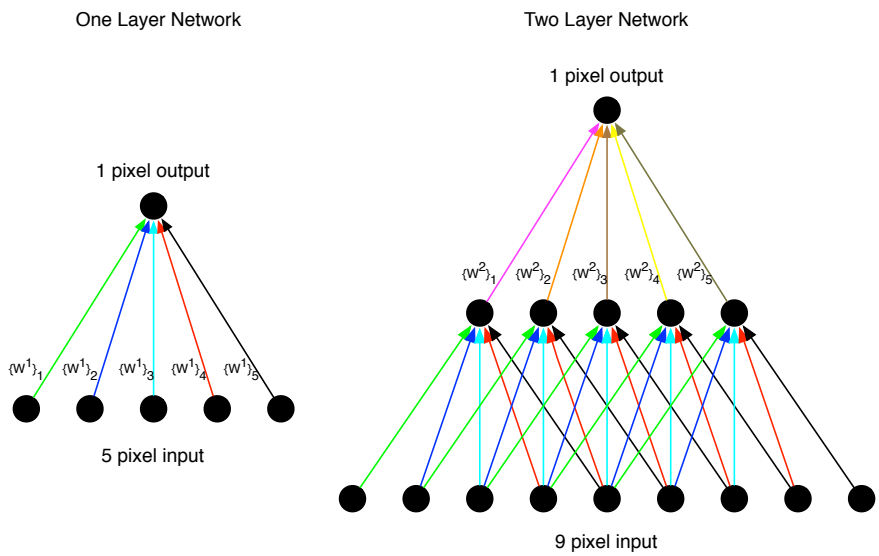


Figure 2-1: Field of view and context in 1d convolutional networks. Each color coded arrow represents a single *scalar* value within a 1d filter. A single output in the one layer network requires 5 pixels, whereas a single output in the two layer network requires 9 input pixels.

> the way the learning architecture uses it. One would like to avoiding overfitting and the need for prohibitively large training sets due to the amount of context used by the algorithm.

Of course, careful consideration of these issues leads to very fundamental questions about vision and image analysis, a full discussion of which exceeds the scope of this work. We focus on how convolutional network use context, and how this compares to some other approaches in the literature.

In object recognition, the amount of context used by an algorithm is the size of the input image it accepts. In image-to-image computations, the total amount of context used to process an image may far exceed the total amount of context used to generate the value of a *single position* in the output image. Hence we define the field of view of an algorithm based on the latter quantity: the total amount of image context used to generate a single output value.

The first question we address is: what determines the field of view of a convolutional network? Consider a convolutional network with no feature maps, just a transformation from input to output specified by a single filter w^1 (see Figure 2-1). Then it is clear that the

field of view of this network depends completely on the pixel (or voxel) dimensions of this filter. There is no opportunity for pixels separated by a distance that exceeds the width of the filter to influence each other in the output. Hence the size of the filters used within the convolutional network are a crucial determinant of field of view.

Now imagine this network has been augmented with an additional layer, sandwiched between the input and output layer. The network is now specified by a filter w^1 that transforms the input into the feature map in this new layer, and another filter w^2 that transforms the feature map in the new layer into the output. The field of view of the network now depends on the size of *both* filters. In particular, the field of view will increase with every additional layer in the network due to repeated convolutions. To see this, it is instructive to "work backwards" by considering the amount of pixel context required to generate a single output value. Clearly, the intermediate layer feature map must be as large as w^2 in order to generate a single output voxel[1]. Thus the input image must be large enough to produce an output value at any location in the intermediate feature map, which requires $|w^2|_i + |w^1|_i - 1$ pixels in the ith dimension, where $|w|_i$ gives the size of filter w in the ith dimension (see Figure 2-1 for an example in the 1d case).

From this simple example we can see that the field of view of a convolutional network can be manipulated by either changing the size of the filters or changing the number of layers in the network. We note that many popular approaches to pattern recognition are based on architectures that are fundamentally "shallow" in the sense of the number of layers (e.g., SVMs). This observation forms the basis of a related issue in the literature of the benefits of using "deep" architectures in machine learning (of which convolutional networks are one example) [42, 4]. This is an issue that is independent of any discussion of context size, but it is also clear that in shallow architectures changing the size of the single set of "filters" that operate on the input is the only option for manipulating field of view.

The second question we would like to address is: how does increasing context through multiple layers compare to using larger filters (in any relevant sense such as efficiency, performance, or ability to be optimized through gradient optimization)? This issue remains

[1] We assume that we perform "valid" convolutions that generate output values only at those locations for which there is sufficient context to apply the entire filter to an input.

incompletely understood, but we offer several conceptual and empirical observations:

1. *Free parameters.* Increasing filter size to enlarge the field of view may increase the number of free parameters within the network architecture more rapidly as compared to increasing the number of layers to achieve the same change in field of view. Consider the case of a shallow network with one filter that has n^k elements (where k is the dimensionality of the image space and n is the length of the filter in a single dimension). Doubling the field of view in all dimensions by increasing the size of the filter will increase the number of free parameters by a factor of 2^k. Alternatively, roughly doubling the field of view in all dimensions by adding another layer to the network with the same size filter increases the number of free parameters by a factor of only 2. This suggests that multi-layer architectures as a strategy for achieving greater field of view may be more resistant to overfitting than large-filter architectures (particularly for higher dimensional image spaces).

2. *Representational power.* A well-known result in the study of multi-layer perceptrons (MLPs) is that in the limit of infinite hidden units, MLPs are universal function approximators [43]. Hence it is not immediately clear why a multi-layer architecture would ever be preferred (to address the issue of context, one could arbitrarily increase the size of the filters used in the initial layer). The benefits of multi-layer architectures may only become clear when the issue of representational power is weighed against the issue of efficiency; a deep architecture may be able to represent a particular computation with significantly fewer overall computations and parameters, as compared to a computationally equivalent shallow implementation [5].

3. *Optimization.* We have already pointed out that the convolutional network architectures we optimize during learning are non-convex and, therefore, we resort to optimization methods such as gradient descent that are only guaranteed to converge to a local minimum. In this context, a fundamentally interesting question is whether certain network architectures are easier to optimize in the sense of generally resulting in configurations of the free parameters with better performance after optimization. We note that while different architectures may have different global minima, ultimately we

are only concerned with performance of minima attainable by the optimization.

Ultimately, it may be difficult to separate the issue of optimization from that of representation; if a particular computation can be implemented more compactly (in terms of overall computations and parameters) by some particular network architecture, then that architecture may also be the easiest to optimize due to the fewer number of parameters. Hence if low-level vision computations fall within that class of computations most efficiently represented by deep architectures, then we might expect optimization of multi-layer convolutional networks to result in better performance on such tasks, as compared to shallow architectures. There is some evidence for this scenario; for example, certain classic hand-designed low-level image processing algorithms, such as anisotropic diffusion, can be viewed as repeatedly performing local computations. This style of computation has a close correspondence to multi-layer convolutional networks in which each layer performs the same computation. Various Markov random field methods can also be interpreted in this way, a connection discussed in much greater detail in the following chapter.

Empirically, we have found that achieving a particular size field of view by increasing the number of layers results in better optimization (lower training error) as compared to using larger filters in a shallow architecture. The reasons for this phenomenon are not well-understood, however, and it may be that a similar minimum is attainable with large filters but requires much longer training times.

Chapter 3

Relationship between Convolutional Networks and Markov Random Fields

Abstract. We discuss the conceptual and mathematical relationship between convolutional networks and Markov random fields (MRFs), a popular approach to image analysis and modeling. This relationship has been briefly mentioned in the literature (see [90]) but never explored in detail.

MRFs are a particular means of probabilistic modeling of images and image processing tasks. The use of a fundamentally probabilistic approach to image analysis has been deemed highly important by some researchers [60, 32, 90], for example because of the desire to maintain an explicit representation of uncertainty and prior information. Elsewhere in this thesis (see Chapters 4 and 5) we compare deterministic methods such as convolutional networks to probabilistic methods such as MRFs in terms of their practical virtues in image processing tasks. In this chapter, however, we emphasize the mathematical connections between two approaches.

We show that certain types of convolutional network architectures can be viewed as performing mean field inference on MRFs. Mean field inference is an approximate inference technique for solving MRFs, and therefore certain convolutional networks can be viewed as performing an approximate and deterministic inference on MRF models. We also discuss how recent trends in increasing the complexity of MRFs have a close correspondence with specific architectural features of a convolutional network.

3.1 Introduction and review of Markov random fields (MRFs)

A Markov random field (MRF) is simply an undirected graphical model. Historically, the term also usually implied a particular conditional independence structure arranged on a lattice to be specialized for image processing, but today the term MRF is used in other contexts as well. Formally, an MRF is specified by a set of random variables $X = \{x_i\}$ that have a Markov property encoded by an undirected graph $G = (V, E)$, where V is a set of vertices, E is a set of edges between vertices, and $i \in V$. We say X is an MRF if it satisfies the following property for all i:

$$p(x_i|X\backslash x_i) = p(x_i|\{x_j\}, j \in N(i)) \tag{3.1}$$

where $j \in N(i) \iff \{i, j\} \in E$. In other words, a random variable $x_i \in X, i \in V$ should be conditionally independent of all other variables given its neighbors $N(i)$ specified by the graph G. We also assume every realization of X has non-zero probability.

Given a distribution of variables that satisfies the conditional property in Equation 3.1, one would like an expression for the joint distribution that exploits the Markov structure. We note that $C \subseteq V$ is a clique iff. $C \subseteq \{i, N(i)\} \forall i \in C$. Then according to the Hammersley-Clifford theorem [9],

$$p(x) = \frac{1}{Z} \prod_{C \in cl(G)} \phi_C(x_C) \tag{3.2}$$

where $cl(G)$ is the set of all cliques of G, Z is a normalization constant, and the functions ϕ are sometimes called the factor potentials or clique potentials. Thus when $p(x) > 0$ (sometimes called the positivity condition), the following are all equivalent:

1. Local Markov property: $p(x_i|X\backslash\{x_i\}) = p(x_i|\{x_j\}, j \in N(i))$
2. Factorization property: $p(x) = \frac{1}{Z} \prod_{C \in cl(G)} \phi_C(x_C)$
3. Global Markov property: $p(x_i|x_j, x_s) = p(x_i|x_s)$ whenever i and j are separated by s (all paths from i to j go through s).

The advantage of using the MRF framework in probabilistic modeling is the ability to easily introduce dependency structures between random variables that reflect prior information about the problem domain. In image processing, the set of vertices and edges in the graph G is typically related to the structure of the discrete space in which the image is embedded (some n-dimensional lattice). For example, in 2d image processing we may have random variables $x_{i,j}$ indexed by locations (i, j) in the image, and edges defined between random variables that are neighbors in the corresponding image space (e.g., $N(i, j) = \{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$). A similar graph can be defined for 3d images: $N(i, j, k) = \{(i \pm 1, j, k), (i, j \pm 1, k), (i, j, k \pm 1)\}$.

Depending on the application, random variables in the distribution may take on binary values (e.g., $x_{i,j} \in \{-1, 1\}$) or integer values (e.g., $x_{i,j} \in [1, 255]$ for encoding of 8-bit intensities).

3.2 Learning and inference in MRFs

Using an MRF for a computational task involves solving the problem of inference, which is broadly defined as inferring the optimal value of some unknown random variables x_U given some observed variables x_O. The specific notion of optimality that is invoked can vary, but a typical choice is the *maximum a-posteriori* (MAP) estimate:

$$x_U^{MAP} = argmax_{x_U} p(x_U|x_O).$$

MAP inference in graphical models is generally an NP-hard problem except in special cases (such as tree-structured directed models, or Ising-like MRFs with non-negatively constrained interaction strengths). Therefore a major focus of probabilistic modeling research is the development of *approximate* inference methods that produce results efficiently, at the expense of accuracy. For a detailed discussion of approximate inference methods in image analysis problems, see [101] or [60]. In this chapter we will focus on the mean field approximation, which is useful for establishing a relationship between convolutional networks and MRFs.

The problem of learning involves choosing the optimal values of some free parameters θ

which are used to control the specific form of the distribution $p(\{x_i\})$, which is sometimes written as $p(\{x_i\};\theta)$ instead. In the supervised learning scenario, both the values of x_U and x_O are known while estimating θ. Thus what remains is to define some criteria for the optimal θ; the most popular choice is the maximum-likelihood criteria:

$$\theta^{ML} = argmax_\theta p(D|\theta) = argmax_\theta \prod_{i=1}^{N} p(d_i|\theta)$$

where $D = \{d_i\}$ is a collection of N "true" samples, each of which can be expressed as a particular configuration of $\{x_U, x_O\}$. This type of learning is sometimes referred to as "generative" due to its focus on maximizing the joint probability of both observed and unobserved data.

Learning is computationally also very difficult, as it involves computing the normalization factor Z in Equation 3.2. Estimating the normalization term involves inference on all possible configurations of the $\{x_i\}$ under the model defined by a specific choice of θ. Effective approximate learning methods are thus a major barrier to success in probabilistic modeling methods; for a detailed discussion see [64, 76].

Note that with the introduction of θ, MAP inference effectively corresponds to $p(x_U|x_O;\theta)$; a fully Bayesian treatment would marginalize out the parameters during inference ($p(x_U|x_O) = \int p(x_U, \theta|x_O)d\theta$) but this is usually very difficult to calculate in practice.

3.3 Image processing with MRFs

In order to formulate a typical image processing task within the framework of probabilistic modeling, we consider a probability density over an "input" image space y, and an "output" density x that is also a distribution over an image space[1]. The precise notion of image space may change depending on the imaging context and the image processing task. For an M-dimensional image with continuously valued intensities we consider $x \in R^M, y \in R^M$. It is often convenient to explicitly denote the topographic organization of the space with some slight abuse of notation: $x \in R^{h\times w}, y \in R^{h\times w}$ where $M = rw$. For binary valued images we

[1] In the terminology introduced in Section 3.2, x can be represented by the unobserved variables x_U and the input y as the observed variables x_O.

may similarly consider e.g., $x \in \{0,1\}^{h \times w}$.

Given the notion of input distribution y and output distribution x, it follows that a basic strategy employed in processing an image with probabilistic modeling is to somehow find x_ξ:

$$x_\xi = argmax_x p(x|y_\xi)$$

where y_ξ is some particular input image drawn from y.

In practice there are at least two different methods for computing x_ξ. In one approach, sometimes referred to as "generative", a model of the joint distribution $p(x,y) = p(x)p(y|x)$ is represented by specifying a prior over the desired interpretation image space, $p(x)$, and an "observation model" $p(y|x)$. Then Bayes rule is used to find the posterior distribution, $p(x|y) \propto p(y|x)p(x)$, which is used during MAP inference.

Another approach, sometimes referred to as "discriminative," is to model the conditional distribution $p(x|y)$ directly. Hence the resulting models are sometimes referred to as conditional random fields (CRFs) [54]. Inference to find x_ξ is performed directly on the specified distribution, without using Bayes rule. Many researchers argue that in practice the discriminative approach is superior because it focuses the model on the distribution truly of interest at inference-time, and thus avoids wasting any modeling power on ultimately irrelevant properties of the joint distribution [53, 59]. Moreover, accurate modeling of even just the prior distribution $p(x)$ may be a significantly more difficult task than modeling the conditional distribution $p(x|y)$. In Chapters 4 and 5 we discuss the relationship between discriminative and generative methods in more detail.

3.4 Mean field theory for MRFs

Mean field theory was developed in the field of statistical mechanics as an approximate means of analytically studying many-body systems for which exact solutions are very difficult to compute due to combinatorial interactions among the random variables [20]. The main goal is to compute the expectation of an arbitrary random variable in a model, an example being

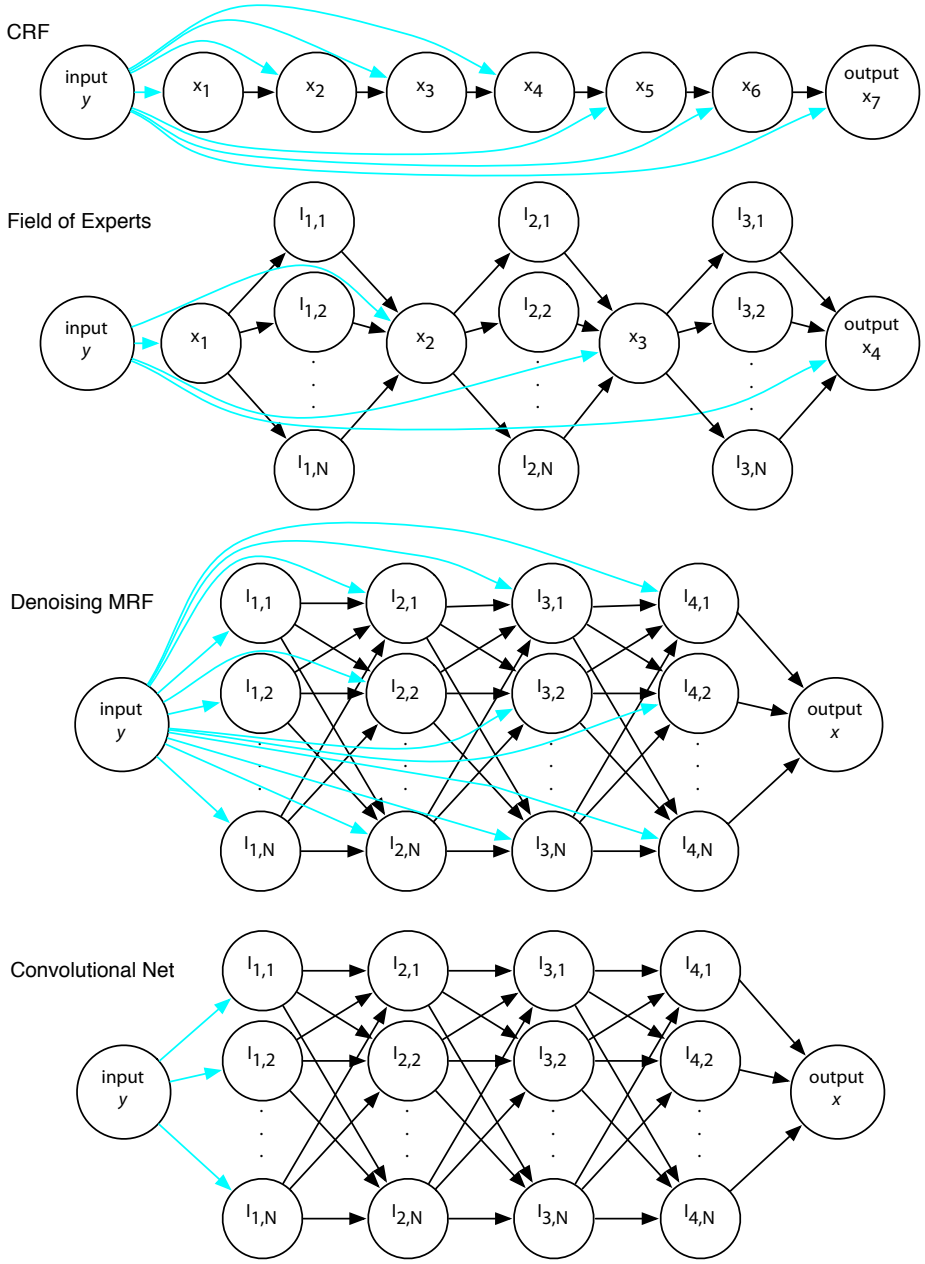


Figure 3-1: Network interpretation of MRFs and a convolutional network architecture. Each node represents an image-valued quantity with identical size as the input image. With certain exceptions (see text for details), arrows denote the operation of convolution followed by a nonlinearity. In all cases they denote influence from one variable to another. Cyan arrows indicate influence from the input image.

$\langle x_i \rangle$ for any i in the model in Equation 3.2:

$$\langle x_i \rangle = \sum_x x_i p(x)$$

where the sum is over all possible configurations of x. However, due to the partition function that appears in computing $p(x)$ and the generally complicated interaction among the various x_i, this sum is computationally intractable.

Therefore, the mean field theory suggests an approximation to this expectation by introducing an assumption: the influence of variable x_j on variable $\langle x_i \rangle$, $j \neq i$, can be approximated by the influence of $\langle x_j \rangle$. In statistical mechanics, the physical justification for this assumption is that at equilibrium, fluctuations at different random variables in the field cancel each other out and can therefore be well-approximated by their expectation. However, in low-dimensions mean field theory can lead to qualitatively incorrect observations about a system [20].

3.4.1 Single feature-map networks and related MRFs

In this section we discuss how convolutional networks with a single feature map in each layer are related to MRFs. In order to do this, we introduce the binary Boltzmann machine, a type of MRF which defines the following density over binary random variables $x = \{x_i\}$:

$$p(x) = \frac{1}{Z} \exp(-\frac{1}{2} \sum_{i,j} W_{ij} x_i x_j + \sum_i b_i x_i) \tag{3.3}$$

where Z is the partition function that normalizes over all configurations of x. An Ising model is a Boltzmann machine that typically has translation invariant-structure in W (e.g., nearest neighbor interactions). Translation invariance is often considered a desirable property of low-level vision algorithms, and translation-invariant MRFs can be expressed in the following way:

$$p(x) \propto e^{\frac{1}{2} \sum_i x_i (w \otimes x)_i + \sum_i b x_i}$$

In this model, the stochastic dynamics of a single variable are given by:

$$x_i \sim \sigma((w \otimes x)_i + b_i) \tag{3.4}$$

Thus an approximation $\mu_i = \langle x_i \rangle$ is given by:

$$\mu_i = \langle \sigma((w \otimes x)_i + b_i) \rangle = \sigma((w \otimes \langle x \rangle)_i + b_i) = \sigma((w \otimes \mu)_i + b_i) \tag{3.5}$$

and thus the entire field μ is given by:

$$\mu = \sigma((w \otimes \mu) + b).$$

These expectations are then thresholded to obtain binary values [60]. A simple way of solving these equations is to iterate them as dynamics:

$$\mu^{t+1} = \sigma((w \otimes \mu^t) + b)$$

where t indexes iterations of the dynamics. At this point, it should be clear that this equation bears a certain resemblance to the convolutional network dynamics defined in Equation 2.1, in which each iteration can be considered a different layer of a network. However, there are several differences:

1. The generic nonlinearity $f(z)$ has been replaced by $\sigma(z) = tanh(z)$
2. Each layer has a single feature map.
3. Each layer uses the same weights (i.e., filter) w.

A more serious issue is that so far we've made no mention of an *input*, which we call y. We have only discussed the prior $p(x)$ over the *desired* image space x. To remedy this, we have two options: introduce an observation model $p(y|x)$, or somehow redefine Equation 3.3 to model $p(x|y)$ directly (the CRF approach). We will take the latter approach, which due to discriminative learning has a closer correspondence to convolutional networks, which

are trained using supervised learning. However, posterior inference for $p(x|y)$ on a model specified by $p(x, y) = p(y|x)p(x)$ would yield a similar relationship.

We consider the conditional random field:

$$p(x|y) \propto e^{\left(\left[\frac{1}{2}\sum_i x_i (w \otimes x)_i + \sum_i x_i (y_i + b)\right]\right)}$$

which is identical to our previous model except that the input has been introduced as an additional bias on the interactions. Hence in the mean field approximation the expectation value μ_i satisfies the equation

$$\mu_i = \sigma\left((w \otimes \mu)_i + y_i + b_i\right)$$

and a dynamics to solve for the expectation is

$$\mu^{t+1} = \sigma((w \otimes \mu^t) + y + b) \tag{3.6}$$

which, for a finite number of iterations, can be considered a convolutional network dynamics with the addition of the image as an input to each layer. Thus certain convolutional network architectures can be seen as performing mean field inference on a CRF.

In a convolutional network, the input image is only provided to the first hidden layer, in which case it can be considered similar to an "initial conditions" to the dynamics above. However, the mean field approximation suggests a different strategy, which is to provide the image as input to each iteration (layer) of computation. This approach is sometimes referred to encoding the image as a "field" to the network. In Figure 3-1 a depiction of the network structure corresponding to this CRF is given; the approximation has been limited to 7 iterations of the dynamics, each of which receives input from the input image.

The dynamics in Equation 3.6 can also be interpreted as a *recurrent* convolutional network because the same weights are used to repeatedly update the interpretation μ of the input y. The individual μ^t are an unraveling-in-time of the recurrent dynamics, a useful strategy for inference because it makes it clear how to apply gradient descent learning algorithms such as backpropogation-in-time.

3.4.2 Multiple feature-map networks and related MRFs

In the previous section, we showed how a convolutional network that has a *single* feature map in each layer and is provided the image as an input to each layer is mathematically related to an MRF. In practice, however, we use convolutional network architectures with *many* feature maps in each layer. In this section, we establish a relationship with such architectures and MRFs.

We consider an MRF recently introduced for low-level image processing known as the "Field of Experts" [91]. This model defines a prior over a *continuously* valued image space $x \in R^M$ in the following way:

$$p(x) = \frac{1}{Z(\theta)} \prod_{k=1}^{M} \prod_{i=1}^{N} \phi_i(J_i^T x_{(k)}; \alpha_i) \tag{3.7}$$

where k indexes over image locations, i indexes over a set of linear filters J_i, α_i is a non-negative scalar, and θ is just the vector of parameters $\{\alpha_i, J_i\}$. Then ϕ_i is termed an "expert" and has the following form:

$$\phi_i(J_i^T x; \alpha_i) = (1 + \frac{1}{2}(J_i^T x)^2)^{-\alpha_i}$$

which was first introduced in MRF modeling by Welling [110] and originally motivated by work on anisotropic diffusion [80, 11]. This prior can be used for image denoising of some noisy input y by adding a pixel-wise Gaussian observation model

$$p(y|x) \propto \prod_{j=1}^{M} e^{(-\frac{1}{2\sigma^2}(y_j - x_j)^2)}$$

which specifies how noisy images (with known standard deviation of noise σ) are generated from clean images x. Then MAP inference can be used to maximize the posterior probability $p(x|y) \propto p(y|x)p(x)$, an approach used by Roth and Black to yield impressive denoising results [91].

Unlike the MRF prior we considered in Equation 5.2, the Field of Experts prior in Equation 3.7 is defined over continuously valued variables. Since the potential functions are

also smooth, it is therefore possible to perform inference by gradient ascent of the posterior probability to find a local maxima of $p(x|y)$. The gradient optimization can be written in the following form [91]:

$$x^{t+1} = x^t + \eta\Big[\sum_{i=1}^{N} J_i^- \otimes \psi_i(J_i \otimes x^{(t)}) + \frac{\lambda}{\sigma^2}(y - x^{(t)})\Big]$$

where J_i^- denotes the filter obtained by mirroring J_i around its center pixel and ψ_i is a non-linearity given by $\psi_i = \frac{\partial}{\partial y} log\phi_i(y; \alpha_i)$.

From this equation we can immediately see an interpretation of inference on $p(x|y)$ as a convolutional network architecture. Computing x^{t+1} involves a convolution of x^t with a set of linear filters, followed by a nonlinearity, to yield an intermediate result: a set of feature maps $\psi_i(J_i \otimes x^{(t)})$. These feature maps are then again linearly convolved with the same set of filters and combined with a field from the noisy input image to yield x^{t+1}. A depiction of this network interpretation is given in Figure 3-1.

Latent within the Field of Experts MRF is thus the notion of multiple feature maps that characterize the convolutional network architectures we use. Yet some significant differences remain: feature maps at every layer share the same set of weights and only interact with each other through a single intermediate image (the estimate x^t of x at some iteration t).

The lack of direct interactions between feature maps is a potential limitation of the Field of Experts MRF model and presents an obvious direction for exploration of more powerful models. We introduce such an MRF, with direct interactions between feature maps and that has been specially designed for image denoising. We show that it is even more closely related to the convolutional network in Figure 4-1 than either the single feature map MRFs studied in Section 3.4.1 or the Field of Experts model. In particular, we consider an MRF that defines a distribution over analog "visible" variables v and binary "hidden" variables h:

$$P(\vec{v}, \vec{h}) = \frac{1}{Z} \exp\left(-\frac{1}{2\sigma^2}\sum_i v_i^2 + \frac{1}{\sigma^2}\sum_{ija} h_i^a w_{i-j}^a v_j + \frac{1}{2}\sum_{ijab} h_i^a w_{i-j}^{ab} h_j^b\right) \tag{3.8}$$

where v_i and h_i correspond to the ith pixel location in the image, Z is the partition function, and σ is the known standard deviation of the Gaussian noise. There are two different notions

of interaction terms in this model; w^a which relates the ath hidden feature map to the visible variables, and w^{ab} which relates the ath hidden feature map to the bth feature map[2]. Hence, $P(v, h)$ constitutes an undirected graphical model which can be conceptualized as having separate layers for the visible and hidden variables. There are no intralayer interactions in the visible layer and convolutional structure (instead of full connectivity) in the intralayer interactions between hidden variables and interlayer interactions between the visible and hidden layer.

From the definition of $P(v, h)$ it follows that the conditional distribution,

$$P(\vec{v}|\vec{h}) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_i\left(v_i - \sum_{ja} w^a_{i-j}h^a_j\right)^2\right) \tag{3.9}$$

is Gaussian with mean

$$v_i = \sum_{ja} w^a_{i-j}h^a_j = \sum_a (w^a \otimes h^a)_i$$

which is also equal to the conditional expectation $E\left[v|h\right]$. We can use this model for denoising by fixing the visible variables to the noisy image, computing the most likely hidden variables h^* by MAP inference, and regarding the conditional expectation of $P(v|h^*)$ as the denoised image. To do inference we would like to calculate $\max_h P\left(h|v\right)$, but this is difficult because of the partition function. However, we can consider the mean field approximation,

$$h^a_i = f\left(\frac{1}{\sigma^2}\sum_j w^a_{i-j}v_j + \sum_{jb} w^{ab}_{i-j}h^b_j\right) = f\left(\frac{1}{\sigma^2}(w^a \otimes v)_i + \sum_b (w^{ab} \otimes h^b)_i\right) \tag{3.10}$$

which can be solved by regarding the equation as a dynamics and iterating it. If we compare this to Eq. 2.1, we find that this is equivalent to a convolutional network in which each hidden layer has the same weights and each feature map directly receives input from the image. A graphical interpretation of this network architecture is given in Figure 3-1.

These results suggest that the convolutional networks we use, with multiple feature maps and interactions directly between feature maps, can be interpreted as performing approxi-

[2]Note that by symmetry we have $w^{ab}_{i-j} = w^{ba}_{j-i}$, and we assume $w^{aa}_0 = 0$ so there is no self interaction in the model (if this were not the case, one could always transfer this to a term that is linear in h^a_i, which would lead to an additional bias term in the mean field approximation).

mate inference on complex MRF models. Computationally, it may be very difficult to work with such a complex MRF; learning and inference even for the simpler models in Equation 3.7 and Equation 5.2 is already difficult and requires extensive use of approximations.

In practice, the convolutional network architectures we train are not exactly related to such MRF models because the weights of each hidden layer are not constrained to be the same, nor is the image an input to any feature map except those in the first layer. An interesting question for future research is how these additional architectural constraints would affect performance of the convolutional network approach.

Finally, although the particular case of Gaussian denoising with a known standard deviation of noise allows for direct integration of the noise model into the MRF equations, our empirical results on blind denoising suggest that the convolutional network approach is adaptable to more general and complex noise models when specified implicitly through the learning cost function. This issue is explored in more detail in the next chapter.

Chapter 4

Natural Image Denoising

Abstract. We investigate a machine learning approach to low-level vision that combines two main ideas: the use of convolutional networks as an image processing architecture and an unsupervised learning procedure that synthesizes training samples from specific noise models. We demonstrate this approach on the challenging problem of natural image denoising. Using a test set with a hundred natural images, we find that convolutional networks provide comparable and in some cases superior performance to state of the art wavelet and Markov random field (MRF) methods. Moreover, we find that a convolutional network offers similar performance in the blind denoising setting as compared to other techniques in the non-blind setting.

4.1 Introduction and related work

This task of image denoising is defined as the recovery of an underlying image from an observation that has been subjected to some noise process. Typically, Gaussian noise is assumed. One approach to image denoising is to transform an image from pixel intensities into another representation where statistical regularities are more easily captured. For example, the Gaussian scale mixture (GSM) model introduced by Portilla and colleagues is based on a multiscale wavelet decomposition that provides an effective description of local image statistics [82, 63].

Another approach is to try and capture statistical regularities of pixel intensities directly

using Markov random fields (MRFs) to define a prior over the image space. Initial work used hand-designed settings of the parameters, but recently there has been increasing success in learning the parameters of such models from databases of natural images [34, 91, 102, 109, 33, 112]. Prior models can be used for tasks such as image denoising by augmenting the prior with a noise model.

Alternatively, an MRF can be used to model the probability distribution of the clean image conditioned on the noisy image. This conditional random field (CRF) approach is said to be discriminative, in contrast to the generative MRF approach. Several researchers have shown that the CRF approach can outperform generative learning on various image restoration and labeling tasks [53, 38]. CRFs have recently been applied to the problem of image denoising as well [102].

The present work is most closely related to the CRF approach. Indeed, certain special cases of convolutional networks can be seen as performing maximum likelihood inference on a CRF, as discussed in Chapter 3. The advantage of the convolutional network approach is that it avoids a general difficulty with applying MRF-based methods to image analysis: the computational expense associated with both parameter estimation and inference in probabilistic models. For example, naive methods of learning MRF-based models involve calculation of the partition function, a normalization factor that is generally intractable for realistic models and image dimensions. As a result, a great deal of research has been devoted to approximate MRF learning and inference techniques that meliorate computational difficulties, generally at the cost of either representational power or theoretical guarantees [77, 100].

Convolutional networks largely avoid these difficulties by posing the computational task within the statistical framework of *regression* rather than density estimation. Regression is a more tractable computation and therefore permits models with greater representational power than methods based on density estimation. This claim is supported by empirical results on the denoising problem, as well as mathematical connections between MRF and convolutional network approaches discussed in Chapter 3.

4.2 Image denoising with convolutional networks

We restrict our experiments to monochrome images and hence the networks contain a single image in the input layer. It is straightforward to extend this approach to color images by assuming an input layer with multiple images (e.g., RGB color channels). For numerical reasons, it is preferable to use input and target values in the range of 0 to 1, and hence the 8-bit integer intensity values of the dataset (values from 0 to 255) were normalized to lie between 0 and 1. We also explicitly encode the border of the image by padding an area surrounding the image with values of -1.

4.2.1 Denoising as a supervised learning problem

The image denoising task must be formulated as a learning problem in order to train the convolutional network. Since we assume access to a database of only clean, noiseless images, we implicitly specify the desired image processing task by integrating a noise process into the training procedure. In particular, we assume a noise process $n(x)$ that operates on an image x_i drawn from a distribution of natural images X. If we consider the entire convolutional network to be some function F_ϕ with free parameters ϕ, then the parameter estimation problem is to minimize the reconstruction error of the images subject to the noise process: $\min_\phi \sum_i (x_i - F_\phi(n(x_i)))^2)$.

Secondly, it is inefficient to use batch learning in this context. The training sets used in the experiments have millions of pixels, and it is not practical to perform both a forward and backward pass on the entire training set when gradient learning requires many tens of thousands of updates to converge to a reasonable solution. Stochastic online gradient learning is a more efficient learning procedure that can be adapted to this problem. Typically, this procedure selects a small number of independent examples from the training set and averages together their gradients to perform a single update. We compute a gradient update from 6×6 patches randomly sampled from six different images in the training set. Using a localized image patch violates the independence assumption in stochastic online learning, but combining the gradient from six separate images yields a $6 \times 6 \times 6$ cube that in practice is a sufficient approximation of the gradient to be effective. Larger patches (we tried 8×8

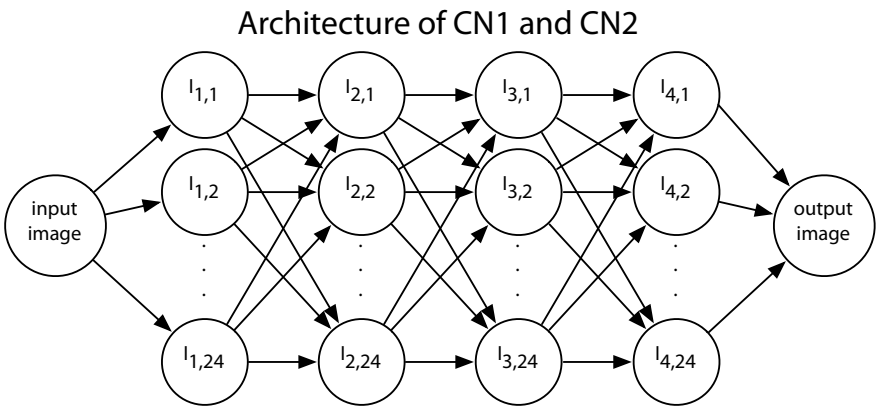


Figure 4-1: Architecture of convolutional network used for denoising. The network has 4 hidden layers and 24 feature maps in each hidden layer. In layers 2, 3, and 4, each feature map is connected to 8 randomly chosen feature maps in the previous layer. Each arrow represents a single convolution associated with a 5×5 filter, and hence this network has 15,697 free parameters and requires 624 convolutions to process its forward pass.

and 10×10) reduce correlations in the training sample but do not improve accuracy. This scheme is especially efficient because most of the computation for a local patch is shared.

We found that training time is minimized and generalization accuracy is maximized by incrementally learning each layer of weights. Greedy, layer-wise training strategies have recently been explored in the context of unsupervised initialization of multi-layer networks, which are usually fine tuned for some discriminative task with a different cost function [41, 86, 4]. We maintain the same cost function throughout. This procedure starts by training a network with a single hidden layer. After thirty epochs, the weights from the first hidden layer are copied to a new network with two hidden layers; the weights connecting the hidden layer to the output layer are discarded. The two hidden layer network is optimized for another thirty epochs, and the procedure is repeated for N layers.

Finally, when learning networks with two or more hidden layers it was important to use a very small learning rate for the final layer (0.001) and a larger learning rate (0.1) in all other layers.

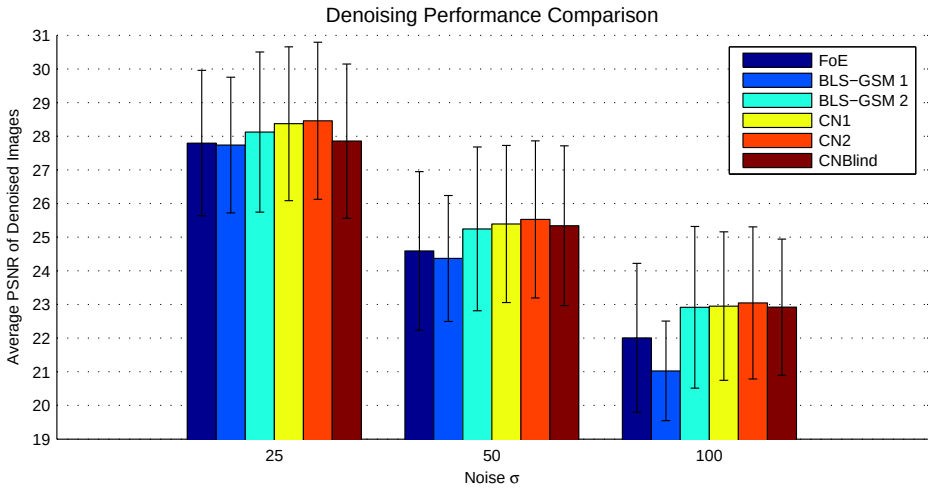


Figure 4-2: Denoising results as measured by peak signal to noise ratio (PSNR) for 3 different noise levels. In each case, results are the average denoised PSNR of the hundred images in the test set. CN1 and CNBlind are learned using the same forty image training set as the Field of Experts model (FoE). CN2 is learned using a training set with an additional sixty images. BLS-GSM1 and BLS-GSM2 are two different parameter settings of the algorithm in [82]. All methods except CNBlind assume a known noise distribution.

4.3 Results

We derive training and test sets for our experiments from natural images in the Berkeley segmentation database, which has been previously used to study denoising [68, 91]. We restrict our experiments to the case of monochrome images; color images in the Berkeley dataset are converted to grayscale by averaging the color channels. The test set consists of 100 images, 77 with dimensions 321×481 and 23 with dimensions 481×321. Quantitative comparisons are performed using the Peak Signal to Noise Ratio (PSNR): $20 \log_{10}(255/\sigma_e)$, where σ_e is the standard deviation of the error. PSNR has been widely used to evaluate denoising performance [82, 91, 63, 102, 109, 33].

Denoising with known noise conditions

In this task it is assumed that images have been subjected to Gaussian noise of known variance. We use this noise model during the training process and learn a five-layer network for each noise level. Both the Bayes Least Squares-Gaussian Scale Mixture (BLS-GSM) and Field of Experts (FoE) method also optimize the denoising process based on a specified noise

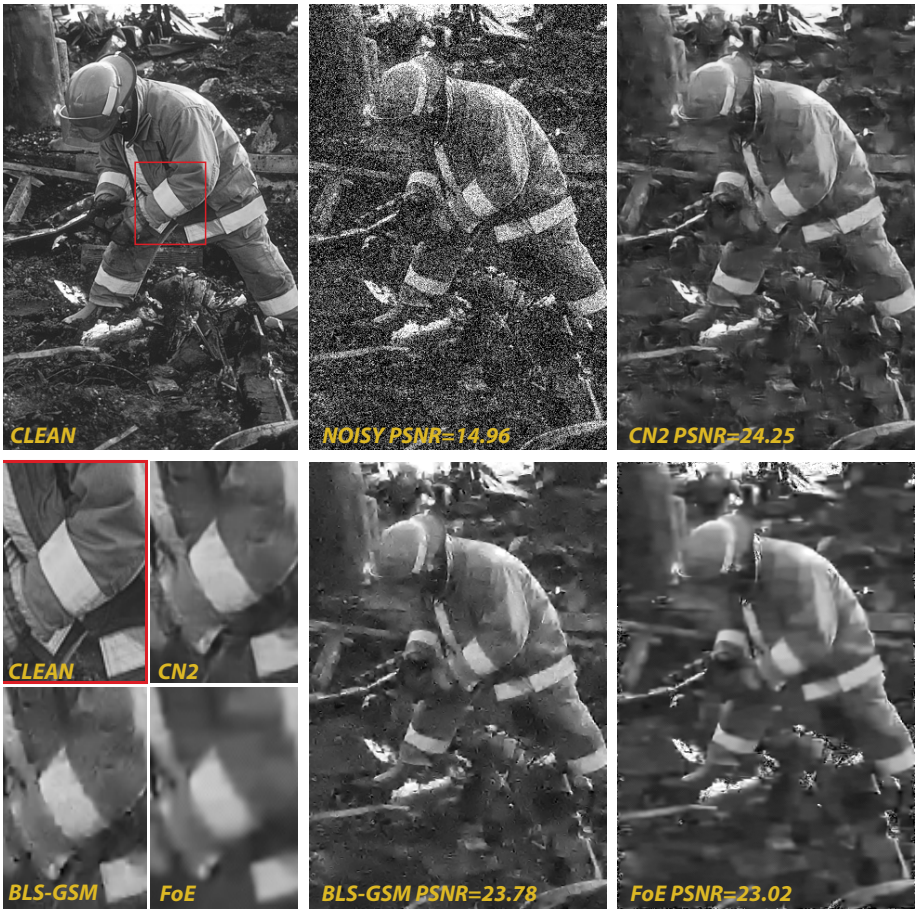


Figure 4-3: Denoising results on an image from the test set. The noisy image was generated by adding Gaussian noise with $\sigma = 50$ to the clean image. Non-blind denoising results for the BLS-GSM, FoE, and convolutional network methods are shown. The lower left panel shows results for the outlined region in the upper left panel. The zoomed in region shows that in some areas CN2 output has less severe artifacts than the wavelet-based results and is sharper than the FoE results. CN1 results (PSNR=24.12) are visually similar to those of CN2.

level.

We learn two sets of networks for this task that differ in their training set. In one set of networks, which we refer to as CN1, the training set is the same subset of the Berkeley database used to learn the FoE model [91]. In another set of networks, called CN2, this training set is augmented by an additional sixty images from the Berkeley database. The architecture of these networks is shown in Fig. 4-1. Quantitative results from both networks under three different noise levels are shown in Fig. 4-2, along with results from the FoE and BLS-GSM method (BLS-GSM 1 is the same settings used in [82] while BLS-GSM 2 is the default settings in the code provided by the authors). For the FoE results, the number of iterations and magnitude of the step size are optimized for each noise level using a grid search on the training set. A visual comparison of these results is shown in Fig. 4-3.

We find that the convolutional network has the highest average PSNR using either training set, although by a margin that is within statistical insignificance when standard error is computed from the distribution of PSNR values of the entire image. However, we believe this is a conservative estimate of the standard error, which is much smaller when measured on a pixel or patch-wise basis.

Blind denoising

In this task it is assumed that images have been subjected to Gaussian noise of *unknown* variance. Denoising in this context is a more difficult problem than in the non-blind situation. We train a single six-layer network network we refer to as CNBlind by randomly varying the amount of noise added to each example in the training process, in the range of $\sigma = [0, 100]$. During inference, the noise level is unknown and only the image is provided as input. We use the same training set as the FoE model and CN1. The architecture is the same as that shown in Fig. 4-1 except with 5 hidden layers instead of 4. Results for 3 noise levels are shown in Fig. 4-2. We find that a convolutional network trained for blind denoising performs well even compared to the other methods under non-blind conditions. In Fig. 4-4, we show filters that were learned for this network.

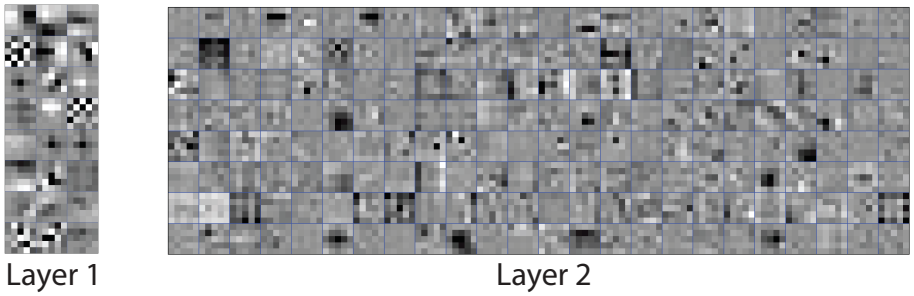


Figure 4-4: Filters learned for the first 2 hidden layers of network CNBlind. The second hidden layer has 192 filters (24 feature maps × 8 filters per map). The first layer has recognizable structure in the filters, including both derivative filters as well as high frequency filters similar to those learned by the FoE model [91, 109].

4.4 Discussion

4.4.1 Prior versus learned structure

Before learning, the convolutional network has little structure specialized to natural images. In contrast, the GSM model uses a multi-scale wavelet representation that is known for its suitability in representing natural image statistics. Moreover, inference in the FoE model uses a procedure similar to non-linear diffusion methods, which have been previously used for natural image processing without learning. The architecture of the FoE MRF is so well chosen that even random settings of the free parameters can provide impressive performance [90].

Random parameter settings of the convolutional networks do not produce any clearly useful computation. If the parameters of CN2 are randomized in *just the last layer,* denoising performance for the image in Fig. 4-3 drops from PSNR=24.25 to 14.87. Random parameters in all layers yields even worse results. This is consistent with the idea that nothing in CN2's representation is specialized to natural images before training, other than the localized receptive field structure of convolutions. Our approach instead relies on a gradient learning algorithm to tune thousands of parameters using examples of natural images. One might assume this approach would require vastly more training data than other methods with more prior structure. However, we obtain good generalization performance using the same training set as that used to learn the Field of Experts model, which has many fewer degrees of freedom.

The disadvantage of this approach is that it produces an architecture whose performance is more difficult to understand due to its numerous free parameters. The advantage of this approach is that it may lead to more accurate performance, and can be applied to novel forms of imagery that have very different statistics than natural images or any previously studied dataset (an example of this is the specialized image restoration problem studied in Chapter 5).

4.4.2 Network architecture and using more image context

The amount of image context the convolutional network uses to produce an output value for a specific image location is determined by the number of layers in the network and size of filter in each layer. For example, the 5 and 6-layer networks explored here respectively use a 20×20 and 24×24 image patch. This is a relatively small amount of context compared to that used by the FoE and BLS-GSM models, both of which permit correlations to extend over the entire image. It is surprising that despite this major difference, the convolutional network approach still provides good performance. One explanation could be that the scale of objects in the chosen image dataset may allow for most relevant information to be captured in a relatively small field of view.

Nonetheless, it is of interest for denoising as well as other applications to increase the amount of context used by the network. A simple strategy is to further increase the number of layers; however, this becomes computationally intensive and may be an inefficient way to exploit the multi-scale properties of natural images. Adding additional machinery in the network architecture may work better. Integrating the operations of sub-sampling and super-sampling would allow a network to process the image at multiple scales, while still being entirely amenable to gradient learning.

4.4.3 Computational efficiency

With many free parameters, convolutional networks may seem like a computationally expensive image processing architecture. On the contrary, the 5-layer CN1 and CN2 architecture (Fig. 4-1) requires only 624 image convolutions to process an image. In comparison, the FoE

model performs inference by means of a dynamic process that can require several thousand iterations. One-thousand iterations of these dynamics requires 48,000 convolutions (for an FoE model with 24 filters).

We also report wall-clock speed by denoising a 512×512 pixel image on a 2.16Ghz Intel Core 2 processor. Averaged over 10 trials, CN1/CN2 requires 38.86 ± 0.1 sec., 1,000 iterations of the FoE requires 1664.35 ± 30.23 sec. (using code from the authors of [91]), the BLS-GSM model with parameter settings "1" requires 51.86 ± 0.12 sec., and parameter setting "2" requires 26.51 ± 0.15 sec. (using code from the authors of [82]). All implementations are in MATLAB.

It is true, however, that training the convolutional network architecture requires substantial computation. As gradient learning can require many thousands of updates to converge, training the denoising networks required a parallel implementation that utilized a dozen processors for a week. While this is a significant amount of computation, it can be performed off-line.

4.4.4 Learning more complex image transformations and generalized image attractors models

In this work we have explored an image processing task which can be easily formulated as a learning problem by synthesizing training examples from abundantly available noiseless natural images. Can this approach be extended to tasks in which the noise model has a more variable or complex form? Our results on blind denoising, in which the amount of noise may vary from little to severe, provides some evidence that it can. Preliminary experiments on image inpainting are also encouraging.

That said, a major virtue of the image prior approach is the ability to easily reuse a single image model in novel situations by simply augmenting the prior with the appropriate observation model. This is possible because the image prior and the observation model are decoupled. Yet explicit probabilistic modeling is computationally difficult and makes learning even simple models challenging. Convolutional networks forgo probabilistic modeling and, as developed here, focus on specific image to image transformations as a regression problem.

It will be interesting to combine the two approaches to learn models that are "unnormalized priors" in the sense of energy-based image attractors; regression can then be used as a tool for unsupervised learning by capturing dependencies between variables within the same distribution [96].

Chapter 5

Binary Image Restoration of Electron Microscopy

Abstract. Convolutional networks are trained using gradient learning to solve the problem of binary image restoration of noisy images. For our training data, we have used electron microscopic images of neural circuitry with ground truth restorations provided by human experts. On this dataset, Markov random field (MRF), conditional random field (CRF), and anisotropic diffusion algorithms perform about the same as simple thresholding, but superior performance is obtained with a convolutional network containing over 34,000 adjustable parameters. When restored by this convolutional network, the images are clean enough to be used for segmentation, whereas the other approaches fail in this respect. We do not believe that convolutional networks are fundamentally superior to MRFs as a representation for image processing algorithms. On the contrary, the two approaches are closely related. But in practice, it is possible to train complex convolutional networks, while even simple MRF models are hindered by problems with Bayesian learning and inference procedures. Our results suggest that high model complexity is the single most important factor for good performance, and this is possible with convolutional networks.

5.1 Introduction and related work

The problem of binary image restoration is defined as the recovery of a "true" binary image from an analog-valued observed image that has been corrupted by some noisy process. The result of image restoration can be used to aid human interpretation, or as the input to other computer vision tasks such as recognition or segmentation.

We test the performance of our approach using a database of electron microscopic (EM) images of neural tissue [18]. It is natural to divide the image pixels into two classes, intracellular ("in") or extracellular ("out"), which is a binary image restoration problem. In order to establish ground truth, as well as provide data for training our convolutional networks, humans manually restored the images by drawing boundaries between "in" and "out" regions. To provide a performance baseline, we first tried restoring the images using simple thresholding, and anisotropic diffusion followed by thresholding. Both methods yielded roughly the same performance.

A convolutional network was then trained on the dataset. Although the network architecture was very complex, containing over 34,000 adjustable parameters, we were able to train it using gradient learning. After training, the network provided significantly more accurate reconstructions on the test set than did thresholding.

For comparison, we also trained a Markov random field (MRF) model on the same dataset. When first introduced to image processing, MRF models were fairly simple, with just a few parameters that were adjusted by hand [34]. Recently there has been significant interest in training more sophisticated MRF models using machine learning methods [91]. Drawing on this research, we trained our MRF using the pseudolikelihood algorithm to generate the noisy training image and the restored training image. Its performance on the test set was not significantly better than simple thresholding.

Some researchers have argued that better results can be obtained from MRF models by training them discriminatively, i.e. by directly optimizing the transformation from noisy to restored images [38, 53]. This is called the conditional random field (CRF) approach. We also trained a CRF on our dataset, but its performance on the test set was no better than the MRF.

To understand the failure of the MRF/CRF approach, we also trained a convolutional network with a very simple architecture. This network could be viewed as mean field inference for a CRF. We added more constraints to the architecture of the network to make it match the CRF as closely as possible. When crippled in this way, the performance of the simple network matched that of the CRF and thresholding.

We suggest that convolutional networks and CRFs should give equivalent performance in principle, as long as the models have equivalent complexity. But our empirical results suggest that a more complex model does better than simpler one. One might worry that a highly complex model should suffer from overfitting of its numerous parameters. However, even with over 34,000 free parameters, our complex convolutional network does not seem to overfit—the gap between training and test error is fairly small. Therefore, high model complexity appears to be an important prerequisite for good image restoration.

While convolutional networks and MRF/CRF models are roughly equivalent in principle, in practice we feel that the former approach is superior because highly complex models can be trained. In the MRF/CRF approach, even training simple models is problematic, because of technical difficulties surrounding Bayesian learning and inference procedures.

We further tested the quality of restorations by using them to generate image segmentations. Since each object within our data is a region of intracellular space separated from all other objects by some extracellular space, a highly clean binary restoration should in principle be sufficient for accurate image segmentation. We demonstrate that the restorations given by convolutional networks can be used to segment the EM images, while the other methods produce restorations that are so error-prone as to be essentially useless for segmentation, at least by naive algorithms.

5.2 Serial Block Face-Scanning Electron Microscopy (SBF-SEM)

A recently developed imaging technique, Serial Block Face Scanning Electron Microscopy (SBF-SEM), is capable of generating nanoscale images of biological tissue over a field of

view spanning up to potentially 1 mm, which is enough to begin reconstructing structures of biological interest [18, 24]. A block of tissue is alternately imaged and sliced, yielding a 3d image with a voxel resolution of 20-30nm in all three directions. Compared to other serial-section imaging techniques, SBF-SEM generates stacks of 2d images in which successive image slices are extremely thin and very well aligned. However, successful interpretation of this data will require accurate and highly automated methods of image processing: a cube of tissue 300 microns on a side could generate a trillion or more voxels of data, making manual analysis so time consuming as to be impractical.

Restoration of SBF-SEM images poses a difficult computer vision problem due to several issues: (1) small size of certain structures (few pixels in diameter at certain points), (2) dense packing of structures, (3) local noise and ambiguity caused by imperfections in the staining and imaging, and (4) variety of physical morphologies.

For the problem of neural circuit reconstruction, a successful image processing approach will have to overcome such issues with *extremely* high accuracy in order to maintain an adequate level of automation. Our basic approach is to restore the images by classifying each voxel as being inside or outside a cell. Since cells are usually separated from each other by some amount of "outside" space, an accurate restoration can provide a segmentation of the dataset as well.

5.3 Creation of the training and test sets

A volume of rabbit retina was imaged at 26.2x26.2x50 nm^3 resolution using the SBF-SEM technique. We used a 20.96x15.72x5μm^3 subset of this volume, yielding a 800x600x100 voxel dataset. The tissue was stained in a manner designed to make image analysis easier for both humans and computers, by attenuating details within the image while enhancing intercellular space [18]. The boundaries of neurons were traced manually by two humans. They provided both object boundaries and consistent object identities from one slice to the next. The tracers were instructed to be careful in drawing boundaries, and generated training data at a rate of roughly 30,000 voxels/hour. An example of an image and its tracing are shown in Figure 5-3. Tracings were captured at a spatial resolution much higher

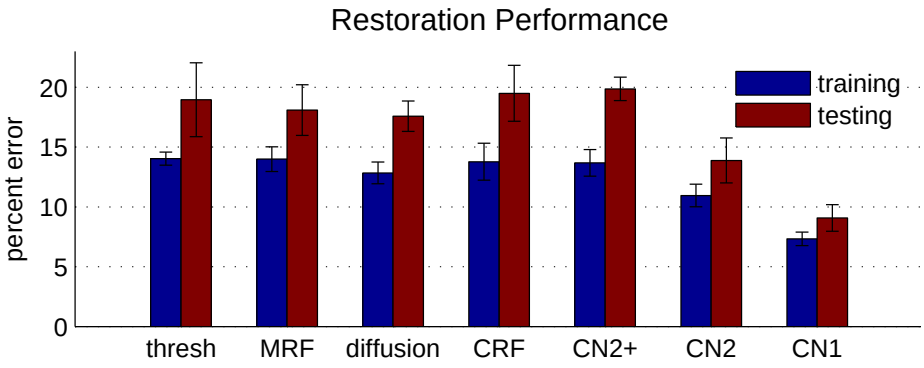


Figure 5-1: Results on voxel-wise restoration accuracy. The training set has 0.5 million voxels, and the test set has 1.3 million voxels.

than the image resolution (the humans traced interpolated images) and were "restored" to an inside/outside binary classification using a point-in-polygon method. Two restorations were generated: one at the same voxel resolution as the images, and one at twice the resolution of the images in each dimension (8 times the number of voxels).

The labeled data was split into two regions: a 0.5 megavoxel "training set" volume that was used to optimize each algorithm, and a 1.3 megavoxel testing region that was used to quantitatively evaluate performance. 75.6% of the voxels within a labeled bounding box were classified as 'inside'. The two human labellings disagreed on the classification of 9.38% voxels within a 1 megavoxel region that was traced by both humans. While this may sound significant, the great majority of inter-annotator differences were found in variations in the exact placement of boundaries, rather than in disagreements over the true shape and identity of objects (i.e., segmentation).

5.4 Results

5.4.1 Thresholding sets baseline performance

Since the "in" regions are usually light and the "out" regions are usually dark, it is natural to attempt binary restoration by simply thresholding the image. The noisy training image was thresholded at various levels to produce a binary restoration, which was compared with the true restoration provided by a human expert. The value of the threshold minimizing error

on the training set was found. Then the noisy test image was thresholded at this value and the result was compared with the human restoration. As shown in Figure 5-1, thresholding yielded a training error of 14.03% and a test error of 18.95%.

An obvious way of improving the simple thresholding algorithm is to preprocess the noisy image by smoothing it. This can be done by linear filtering, but there are also more powerful "edge-preserving" nonlinear techniques that smooth differences in image intensities except at regions of very strong discontinuity. We used a 3d version of the Perona-Malik anisotropic diffusion algorithm [80]. Binary restorations were produced by thresholding the diffused images. The threshold, along with several parameters of the diffusion algorithm, were optimized by grid search on the training set (see supplementary material for additional details).

Thresholding the diffused images did not yield significantly better results than thresholding the raw images. This may be due to the fact that the inside regions of cells were not of uniform intensity. Although "in" regions were generally lighter than "out" regions, some "in" regions were still fairly dark. Anisotropic diffusion smoothed the "in" regions but did not change their average intensity. Therefore some "in" regions still fell below threshold and were erroneously classified as "out."

5.4.2 A complex convolutional network outperforms simple thresholding

A convolutional network alternates between linear filtering and nonlinear transformations to produce a transformed version of some input. The architecture consists of an input layer that encodes one or more input images, an output layer that encodes one or more output images, and various intermediate layers with "hidden" images that contain the internal computations and representations of the algorithm. Each layer receives input from only the previous layer. The activity of feature map a in layer k is given by

$$I_a^k = f\left(\sum_b w_{ab}^k \otimes I_b^{k-1} - \theta_a^k\right) \tag{5.1}$$

where the I_b^{k-1} are feature maps in the previous layer that provide input to I_a^k, and $\otimes$ denotes the convolution operation. The function f is a smooth nonlinearity; we use the sigmoid $f(x) = 1/(1+e^{-x})$. There is also a threshold parameter θ_a^k associated with each feature map.

We trained a convolutional network of the architecture shown in Figure 5-2 on our dataset of EM images. All filters and biases in the network were optimized using an online version of the backpropagation algorithm (see supplementary material for further details). We used the cross-entropy cost function as our optimization criterion. The network was trained for 100 epochs, where one epoch is defined as exposure to the entire training set of 0.5 megavoxels. This took 48.7 hrs, using a parallel implementation running on 12 cpu-cores operating at 3Ghz. While this may seem significant, it should be noted that our network is a highly complex model by the standards of the image processing field, having over 34,000 adjustable parameters.

Since the output of the convolutional network is analog, it was thresholded to produce a binary image restoration. A threshold value of 0.51 was chosen by optimizing restoration on the training set. The training and test errors are shown in Figure 5-1. The convolutional network roughly halved the error rate of simple thresholding or thresholded anisotropic diffusion, which was a statistically significant improvement. Visual inspection also shows obvious improvement (Figure 5-3).

5.4.3 Empirical comparison to MRFs

5.4.3.1 MRF performance is similar to thresholding

A conventional approach to image restoration is Bayesian inference and learning using Markov random field (MRF) models [60]. We also applied this approach to our dataset. Let $y = \{y_i\}$ denote the observed image and $x = \{x_i\}$ denote the true image, where i ranges over all voxel locations. The joint density $p(x, y)$ is specified by a prior distribution $p(x)$ over the true image x, and a noise model $p(y|x)$ that describes how noisy images are generated from

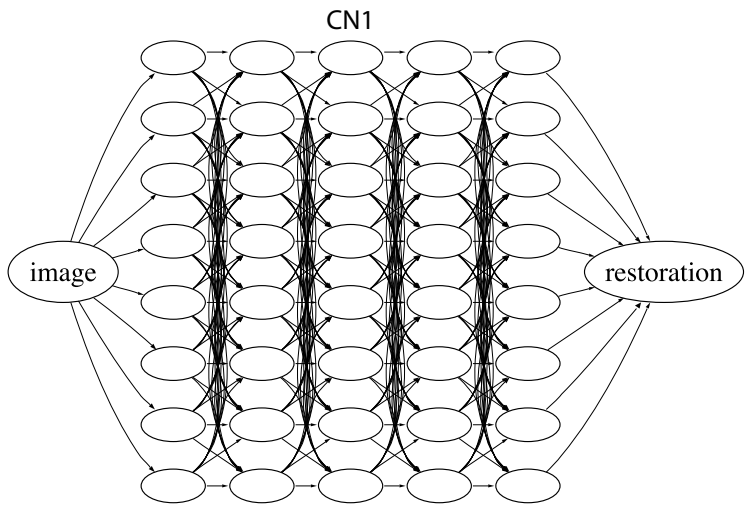


Figure 5-2: Complex convolutional network architecture (CN1). Between the input image and output image are 5 hidden layers, each containing 8 images. All arrows represent adjustable $5 \times 5 \times 5$ filters, and each node has an adjustable bias parameter. Because there are six convolutions between the input and output, each voxel in the output image is a function of a $25 \times 25 \times 25$ patch of the input image. The total number of free parameters is 34,041.

true images. We considered a prior of the MRF form,

$$p(x) \propto e^{\frac{1}{2}\sum_i x_i (w \otimes x)_i + \sum_i b x_i} \tag{5.2}$$

where the x_i are binary variables taking on the values ± 1. Since the MRF is assumed to be translation invariant and local, the interactions between voxels are described by a filter w of size much smaller than the image. We used a $5 \times 5 \times 5$ filter, which permits interactions up 2 pixels away in all directions. The filter is invariant under reflections, $w_i = w_{-i}$, and its central voxel is assumed to be zero (no self-interaction). We used the Gaussian noise model

$$p(y_i|x) = p(y_i|x_i) \propto \sum_{l=\{-1,1\}} \delta(x_i, l) e^{-\frac{(y_i - \xi_l)^2}{2\sigma^2}} \tag{5.3}$$

where $\xi_{\pm 1}$ are the means of "in" and "out" voxels, and σ is their standard deviation.

The model was trained on the images by maximizing

$$\log p(x, y) = \log p(y|x) + \log p(x) \tag{5.4}$$

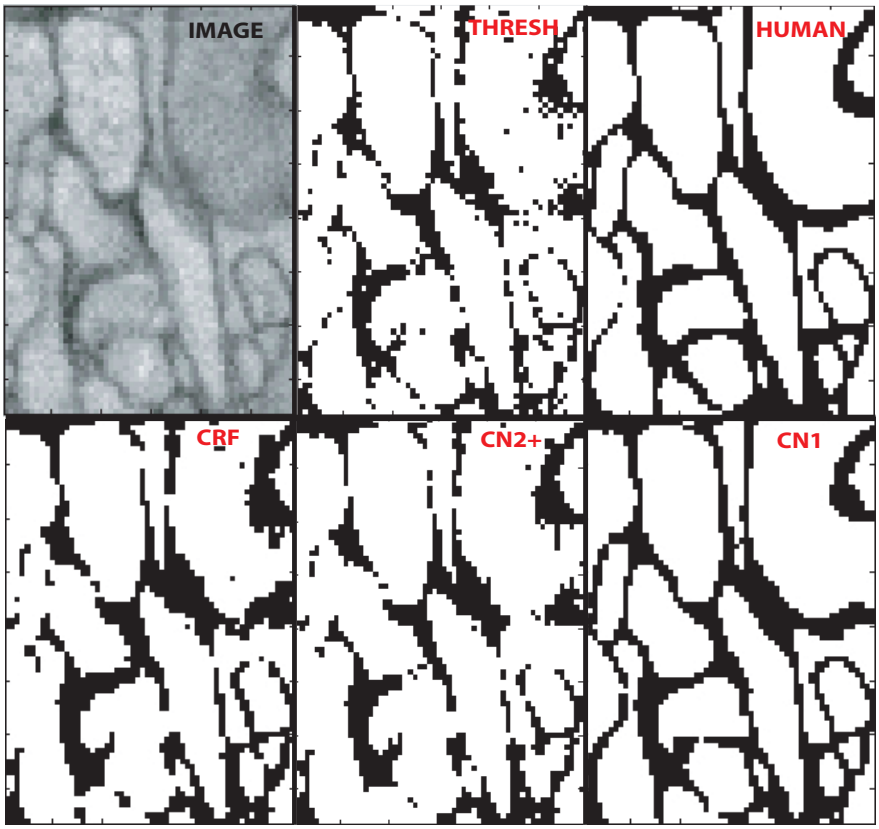


Figure 5-3: Example of restoration results from the test set (generalization performance). Although only a 2d region is shown here, all methods except thresholding utilize 3d computations.

with respect to the adjustable parameters, where y was the noisy SBF-SEM image and x was the restored image from the training set. Since the two terms in the sum do not share parameters, they can be optimized independently. The parameters $\xi_{\pm 1}$and σ of the noise model were found by calculating the mean and standard deviation for "in" and "out" voxels of the training data. The parameters of the MRF prior $p(x)$ were determined by pseudo-likelihood learning, which has become popular in MRF research [77, 60, 10, 38].

Once these parameters were trained, we attempted to restore the noisy test image by maximizing the posterior distribution $p(x|y)$ with respect to x, a procedure known as MAP inference. The posterior distribution $p(x|y)$ takes the same form as Eq. (5.2), except that b is replaced by a quantity that depends on the noisy image y.

For MAP inference, we first attempted simulated annealing via MCMC sampling on the posterior distribution. However, because of its slow convergence, we could not be sure

whether the global optimum was attained. Better results were obtained with the min-cut algorithm of Boykov and Kolmogorov (BK) [16, 17], which has become a popular alternative to sampling. Although the BK min-cut algorithm is fast, it is rather memory-intensive, storing an adjacency graph which in our case was 125 times the size of the already large image. Therefore, the bounding box of the test image was split into four overlapping regions, each of which contained roughly 1.6 million voxels. Then min-cut was run on each region, and the results were stitched together. A further complication is that min-cut is only guaranteed to give the global optimum when the interaction strengths w are nonnegative, but pseudolikelihood training yielded interaction strengths of mixed sign. Nevertheless, min-cut provided superior results to simulated annealing in practice, so its results are shown in Figure 5-1.

The MRF model did not perform significantly better than simple thresholding. We can only speculate about the reasons for the failure of sophisticated Bayesian methods. First, the pseudolikelihood training procedure might have been problematic. Perhaps true maximum likelihood learning of the MRF prior $p(x)$ would yield better performance. In principle this could be done using the Boltzmann machine learning algorithm [1], but this would have required MCMC sampling, an extremely time-consuming learning procedure on a 0.5 million voxel training region. Second, the min-cut inference procedure might not have given the global optimum, since the interaction strengths were of mixed sign. Third, it might have been misguided to maximize the joint probability $p(x, y)$, a strategy known as generative training. The desired computational task is not generating pairs of noisy and restored images, but rather to transform a noisy image into a restored image. Therefore, it might be better to directly maximize the probability $p(x|y)$, a strategy known as discriminative training. In fact, there is a great deal of current research on MRF models motivated by the belief that discriminative training is superior to generative training.

5.4.3.2 CRF performance is similar to thresholding

Discriminatively trained MRFs are sometimes called conditional random fields (CRFs) [54, 53, 38, 28]. Some have argued that CRFs are superior to MRFs for image analysis [38, 53, 84]. To test this claim, we trained a CRF model of the form

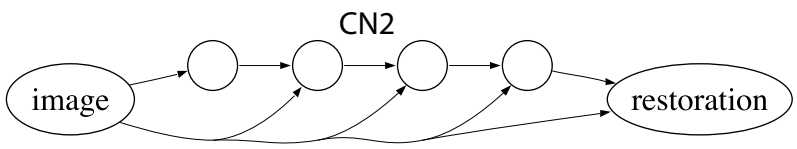


Figure 5-4: Simple convolutional network with architecture matched to the MRF model used in this paper. A single $5 \times 5 \times 5$ filter and bias are repeated five times. Each layer receives input from the raw image, from which an offset parameter is subtracted. The filter, the bias, and the offset give 127 total adjustable parameters.

$$p(x|y) \propto e^{\left(\beta\left[\frac{1}{2}\sum_i x_i (w \otimes x)_i + \sum_i x_i (y_i + b)\right]\right)} \tag{5.5}$$

We attempted both pseudolikelihood and zero temperature Boltzmann machine learning, which is valid in the limit $\beta \to \infty$. The latter gave superior results. In this approach, min-cut was used to find the global optimum x^* of the CRF at each iteration. Then the contrastive update

$$\Delta w_j \propto \sum_i (x_{i+j} x_i - x^*_{i+j} x^*_i) \tag{5.6}$$

was made, where x denotes the human-restored image. A similar update rule was used for the offset parameter b in Eq. (5.5). Since min-cut is only guaranteed to work for nonnegative filters, whenever the contrastive update drove a filter coefficient negative, it was reset to a value of zero. Once the training had converged, min-cut was used to restore the test image. Again, the test error was not significantly better than that of simple thresholding, as shown in Figure 5-1.

5.4.4 Failure analysis of MRFs

We were surprised that neither the MRF nor CRF models were significantly better than thresholding. How could these empirical results be explained? To explore this issue, we decided to train a convolutional network that was much simpler than the one of Figure 4-1. Figure 5-4 depicts the architecture of CN2. We chose the architecture because it can be viewed as mean field inference on the CRF of Eq. (5.5). Additionally, we restricted the weights in w to be positive, to match the constraint in the min-cut based Boltzmann learning

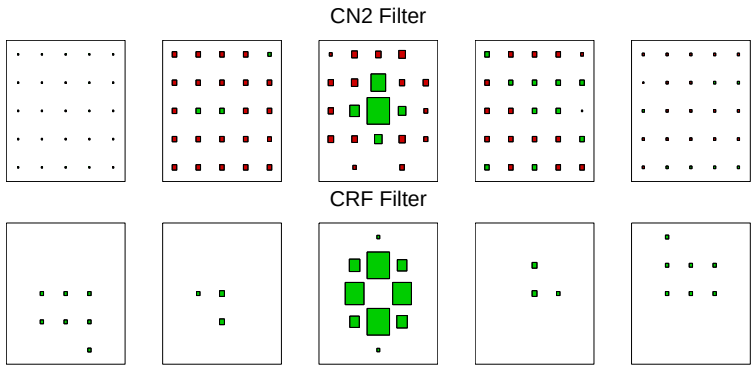


Figure 5-5: Filters learned by CN2 and the CRF. Each box displays one layer of weights. Green and red boxes signify positive and negative weights, respectively, while size indicates strength. Our results show that the negative surround in CN2 is important for good image restoration. Both the CRF and CN2$^+$filter were constrained to be nonnegative, which yielded poor performance.

procedure employed for the CRF.

The performance of this network, called CN2$^+$, was nearly identical to the CRF in performance (Figure 5-1). When inspected visually, the CN2$^+$and CRF restorations were similar (Figure 5-3). This is consistent with the idea that convolutional networks and CRFs are closely related to each other. When the models are exactly matched, they should yield roughly the same performance.

But we hypothesized that a more complex model should yield better performance than a simpler model. To test this idea, we relaxed the non-negativity constraint of CN2$^+$. This network, called CN2, was significantly better than thresholding, the MRF, and the CRF, but not as good as the complex CN1 network of Figure 4-1. The weights of the CN2 and the CRF are compared in Figure 5-5. CN2 has a positive center and a negative surround, suggesting that CN2 outperforms CN2$^+$ and the CRF because of its negative filter coefficients. This example shows that a seemingly trivial increase in the representational capability of a model can lend it superior performance.

5.5 Learning super-resolution restoration by upsampling

The nature of SBF-SEM images enables a simple approach to image segmentation based on binary image restoration. We tested this by restoring the images and then using a connected

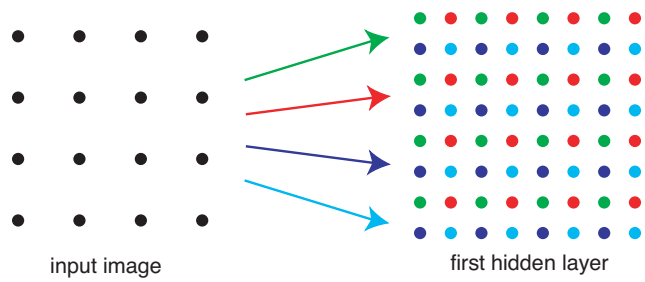


Figure 5-6: The first layer of CN3, a super-resolution convolutional network, produces a 2× oversampled restoration relative to the input image by convolving 8 filters on 1 location, which are interleaved to form a single, oversampled image. This is illustrated here for the 2d situation, in which there are 4 filters for 1 location.

components algorithm to produce distinct image domains. Although more sophisticated segmentation algorithms might be considered, this simple procedure establishes a lower-bound on relative segmentation performance between the restorations.

A complication in this strategy arises when two objects are so close together that a binary restoration contains no boundary between them. In this case, even a completely accurate restoration may merge the two objects under a connected components criteria. Thus, we generate restorations at a *super-sampled* resolution twice that of the original images. If a human expert has decided that two objects are distinct despite the lack of a local boundary, the super-sampled representation allows enough room in the image space for there to be out-voxels between the objects. This was confirmed by segmenting the supersampled human restorations using the connected components criteria.

A convolutional network, CN3, was designed to produce supersampled restorations from the input images. The first layer of the network performs an upsampling by having 8 filters that each look at the same location in the input image, but output a different voxel within the $2 \times 2 \times 2$ cube that each input position now generates (Figure 5-6). Upsampling is then followed by an additional 4 layers of processing.

For the MRF and diffusion results, we manually upsampled the images using a linear interpolation scheme (other interpolation methods were also tested). However, the simplicity of convolutional networks allows us to easily *learn* the transformation from a "1×" image to a "2×" upsampled restoration.

The results clearly demonstrate the benefit of the convolutional network approach (Figure

5-7). The restoration from MRF and diffusion based methods contained many low-level errors, particularly in difficult image locations at which two objects were likely to merge together or one object was likely to break apart. Consequently they produced segmentations that were basically useless, even on the *training* set. The thresholded diffusion segmentation lacks many objects and the shapes of those that are present are severely distorted. As for the CRF, almost all objects were merged together into one large blob, due to inadequately preserved object boundaries in the CRF restoration. In contrast, the convolutional network restoration has far fewer errors and thus the segmentation is far more reliable, although not perfect.

5.6 Discussion

5.6.1 Convexity comes at the cost of representational power

In general, the problem of MAP inference for the MRF model (5.2) is an NP-hard combinatorial optimization. But for the special case of nonnegative interaction strengths, MAP inference is equivalent to a network flow problem, which is a convex optimization [35]. This realization has led to a great deal of exciting progress in the form of min-cut/max-flow algorithms for MAP inference [16, 17]. However, researchers may have overlooked the fact that the nonnegativity constraint might compromise the representational capability of their MRF models. In our empirical tests, the CRF model performed no better than the naive algorithm of thresholding, and worse than CN2, which can be viewed as an approximate inference algorithm for the CRF. The reason for the inferior performance of the CRF appears to be the nonnegativity constraint, which was imposed as a requirement for the min-cut algorithm. Our evidence for this comes from the fact that the performance of $CN2^+$,which was just like CN2 but with a nonnegative filter, dropped to the same level as the CRF and thresholding. Although negative interaction strengths are important for good performance (Figure 5-5), they are not allowed if convex optimization methods are used. More generally, while researchers may be attracted by the prospect of efficient algorithms for convex optimization, they should also be wary that convexity could come at a cost.

Convolutional networks avoid many of the technical problems of MRFs As a representation for image processing algorithms, convolutional networks have many of the same virtues as MRFs. Mathematically, the two approaches are closely related: certain convolutional networks can be viewed as mean field inference for discriminatively trained MRFs. However, convolutional networks avoid the technical problems of Bayesian learning and inference that were described above. The gradient of the objective function for learning can be calculated *exactly*, while Bayesian methods of calculating the gradient of MRF likelihoods rely on approximations.

Convolutional network learning is founded not on Bayesian principles but rather on the principle of empirical error minimization [106]. This principle is simple and direct: find a member of the parametrized function class defined by convolutional networks of a given architecture by minimizing error on the training set.

The primary drawback of empirical error minimization is the requirement of databases with labeled examples. Creation of such databases may require substantial labor, particularly in image processing applications. However, as the goal of many low-level vision algorithms lack a robust mathematical specification, the creation of labeled datasets may be the only way to rigorously evaluate and optimize algorithms [68]. Moreover, recent advances in unsupervised learning in neural networks may dramatically reduce the amount of labeled data that is required to achieve good generalization performance [41, 74].

5.6.2 Pseudo-likelihood can be a poor approximation

True maximum likelihood learning for MRF models depends on MCMC sampling methods. Because these methods can be time-consuming, researchers have looked for approximate methods that are faster, such as the popular pseudolikelihood (PL). But when we used PL to train a CRF, the results were so poor that they were not worth reporting. This is likely because the PL procedure trains the CRF to predict the value of a single output voxel from the values of other output voxels. Evidently this task is not so relevant for predicting output voxels based on the input image. There are other approximate learning procedures, such as contrastive divergence [77, 41], but they are not guaranteed to give good results either. For the simple MRF model (5.2) one can imagine that MCMC sampling methods

would eventually work, given enough computational time. But ML training of an MRF with complexity comparable to CN1 (Figure 4-1) would be even more difficult. One would be forced to use approximations of questionable accuracy, much like pseudolikelihood. In short, Bayesian inference and learning procedures are beset with technical difficulties. One can argue about whether it is worth putting in the effort to surmount these difficulties, but it is difficult to deny that they exist.

5.6.3 Discriminative training is not always better than generative training

In our particular application, a discriminatively trained CRF did not give better results than a generatively trained MRF; both were about the same as thresholding. A possible explanation is that our CRF/MRF model is such an impoverished representation that it does not matter whether discriminative or generative training is used (though it should be noted that the use of a learned $5 \times 5 \times 5$ filter makes our CRF/MRF model richer than many studied in the literature). Perhaps if a more complex CRF/MRF model were trained, there would be more of a difference between discriminative and generative training, but this speculation remains untested.

A convolutional network makes use of context to do image processing As mentioned in the introduction, our work is closely related to neural networks that operate on image patches, which have been studied extensively [27]. A convolutional network can be regarded either as a generalization or as a special case of an image patch network. Both viewpoints will be described briefly here, starting with the former.

The filters of CN1 were all 5^3 (Figure 4-1). If the filters were reduced in size to 1^3 after the first 5^3 layer, then CN1 would be equivalent to a neural network that is applied to each 5^3 image patch. This means that CN1 can be seen as a generalization of an image patch network.

But in fact, each 5^3 convolution in CN1 increases the size of the input patch that is "seen" by a voxel. In the final output layer, a voxel depends on a 25^3 patch of the input image, because of the six intervening convolutions. Consequently, CN1 uses a much larger context

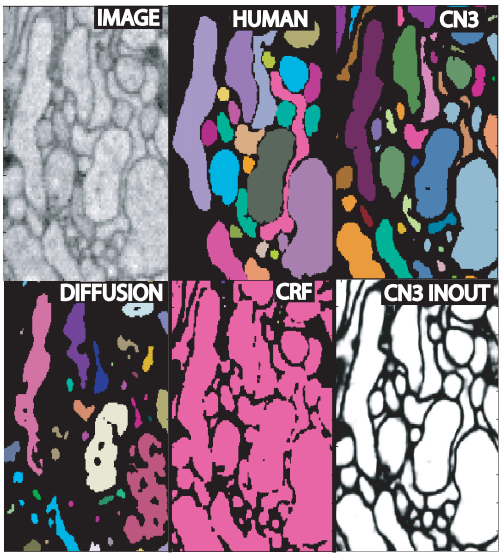


Figure 5-7: Example of segmentation results on the test set. Diffusion and CRF segmentations are poor due to many errors in the restoration. CN3's superior output ("CN3 inout" shows pre-thresholded restoration) provides a more reliable segmentation.

than 5^3 to compute the value of each output voxel.

A convolutional network makes efficient use and reuse of computation Alternatively, CN1 can be seen as a special case of a neural network that takes 25^3 patches as input, in which the synaptic weights are constrained to have a convolutional structure. This constraint makes CN1 highly efficient in its use of computational resources. Consider two 25^3 image patches that are displaced by a single voxel. If a general neural network were applied to both patches, the computations would be completely separate. But for CN1, most computations are shared by neighboring image patches, suggesting that convolutional networks are highly efficient algorithms for image processing.

5.7 Specific Methods

5.7.1 Evaluation Methodology

Training set voxel-wise restoration accuracy was found by measuring the percentage of voxels that agreed with a binary human classification. For the convolutional networks, we used the restoration provided by the training epoch at which convergence was observed (i.e., little to

no fluctuation in training error for several successive epochs). For methods whose outputs are real-valued, such as the convolutional networks or diffusion, a threshold was applied to produce a binary classification. The threshold value was chosen to maximize accuracy on the training set.

Testing set performance was measured by preserving all parameters, threshold values, etc., from training set optimization and then measuring voxel agreement on a separate part of the image that had been manually classified by the same human as for the training set.

Individual samples in voxel-wise accuracy measurement cannot be considered to truly satisfy independent and identically distributed (IID) assumptions due to correlations resulting from a contiguous image space. Therefore, the training and test areas were divided into ten non-overlapping regions, and accuracy was then measured within each region. The standard error of measurement was then calculated from the variance in accuracy among these regions. Although these image regions are not truly statistically independent, we expect the correlations in the accuracy measurements of these regions to be relatively weak, so that the standard error of measurement is a reasonable estimate of the error bar.

5.7.2 Procedures

5.7.2.1 Convolutional networks

The filters w and thresholds θ are free parameters of the algorithm that are chosen by gradient learning, using a version of the backpropagation algorithm adapted to multi-layer convolutional networks [55]. We used the cross-entropy cost function as our optimization criterion.

A stochastic, online gradient learning procedure was found to be more efficient than batch training (see [13] for an interesting study of the virtues of online learning methods). Each training epoch consisted of a random sequence of $6 \times 6 \times 6$ cubes chosen from the training set. The network parameters were then updated after being evaluated on each cube in the sequence. This learning procedure was highly reliable; larger cube sizes decreased training speed but did not improve training quality. The learning rate of each free parameter was controlled by a stochastic approximation to the diagonal of the hessian matrix [57].

Before learning, a weight in filter w_{ab}^k was initialized to a random value chosen from a normal distribution with zero mean and a variance inversely proportional to the square root of the number of voxels in the filter (e.g., $\frac{1}{\sqrt{125}}$for a $5 \times 5 \times 5$ filter). Although such details were not critical to the success of the learning procedure, in practice they were found to speed convergence rates.

5.7.2.2 MRF/CRF Models

Inference

Inference with our MRF and CRF models was done by maximizing the posterior distribution $p(x|y)$ with respect to x, which is known as the maximum a posteriori (MAP) estimate.

For finding the MAP with min-cut [16, 17] we constructed a graph with the nonnegative adjacency weights given by w and source/sink connections given by $\max\{0, +\tilde{b}_i\}$, $\max\{0, -\tilde{b}_i\}$ respectively, where

$$\tilde{b_i} = b + \frac{\xi_1 - \xi_{-1}}{2\sigma^2}\left(y_i - \frac{\xi_1 + \xi_{-1}}{2}\right) \tag{5.7}$$

for the MRF; and

$$\tilde{b_i} = y_i + b \tag{5.8}$$

for the CRF. To compute the cut we employed the code provided with [16].

We also tried simulated annealing with a fast annealing schedule for finding an approximation to the MAP solution [60, 47]. At each time step, the new value for a randomly chosen variable x_i was sampled from the conditional distribution

$$P(x_i|x_{-i}, y_i) \propto e^{x_i\left(\{w \otimes x\}_i + \tilde{b_i}\right)/T} \tag{5.9}$$

where x_{-i} denotes all variables except for x_i, and T is the temperature. The initial value for the temperature was $T = 10$. This value was decreased by $1/20$ each timestep for 80 steps. The MAP estimated by simulated annealing was usually very similar to the MAP computed with min-cut.

MRF Learning

The updates for maximum likelihood estimation of the parameters w and b for the prior are given by Boltzmann machine learning updates [1],

$$\Delta w_j \propto \left\langle \sum_i x_{i+j} x_i \right\rangle_0 - \left\langle \sum_i x_{i+j} x_i \right\rangle_\infty \tag{5.10}$$

and

$$\Delta b \propto \left\langle \sum_i x_i \right\rangle_0 - \left\langle \sum_i x_i \right\rangle_\infty \tag{5.11}$$

where $\langle \cdot \rangle_0$ is the average with respect to the empirical distribution given by the data and $\langle \cdot \rangle_\infty$ is the average with respect to the model distribution given by the parameters w and b. Because computing the average with respect to the model distribution requires sampling the entire configuration space it is intractable for the dimensions of our data. We therefore applied pseudolikelihood learning [10] which estimates the model distribution average by the mean field generated by the data,

$$\left\langle \sum_i x_{i+j} x_i \right\rangle_\infty \approx \left\langle \frac{1}{2} \sum_i \left(x_{i+j} \sigma_i + \sigma_{i+j} x_i \right) \right\rangle_0 \tag{5.12}$$

and

$$\left\langle \sum_i x_i \right\rangle_\infty \approx \left\langle \sum_i \sigma_i \right\rangle_0 \tag{5.13}$$

where $\sigma_i = \tanh\left((w \otimes x)_i + b \right)$ is the conditional $P(x_i | x_{-i}, y_i)$.

We ran the update 400 times with a learning rate of 0.1.

CRF Learning

The Boltzmann updates for CRF learning are the same as for the MRF. However, in CRF learning we replaced the model distribution average by the zero temperature limit computed by the min-cut algorithm. We used 400 updates and a learning rate of 0.01.

5.7.2.3 Anisotropic diffusion

We applied a 3 dimensional version of Perona-Malik anisotropic diffusion to the image, that iterates the following discretized update to image x at each position i [80] :

$$X_i^{t+1} = X_i^t + \frac{\lambda}{|\eta_i|} \sum_{j \in \eta_i} g(\nabla X_{i,j}^t) \nabla X_{i,j}^t \tag{5.14}$$

where η_i denotes a neighborhood for pixel at position i, t indexes discrete time steps, λ is a diffusion rate, $\nabla X_{i,j}^t = X_i^t - X_j^t$ and g is the edge-stopping function:

$$g(x) = \frac{1}{1 + \frac{x^2}{K^2}}. \tag{5.15}$$

Parameters λ, K, the number of iterations, and the threshold value were optimized by grid search to maximize classification accuracy on the training set. We also tried coherence-enhancing diffusion [108], which on this dataset yielded very similar results to Perona-Malik.

5.7.2.4 Segmentation

Segmentations were produced from binary restorations using a simple 3d connected-components algorithm: the restoration was converted into an undirected graph, in which each voxel of the image is a node in the graph and edges are inserted between two nodes when their corresponding voxels are (1) direct neighbors along any of the principle axes in the 3d image space (i.e., a 6-connected neighborhood) and (2) share the same binary restoration value. Segmented domains are then produced by searching for connected components of this graph.

Chapter 6

Digital Topology and Image Segmentation

Abstract. A major claim in this thesis is that by adopting concepts from digital topology, we can rigorously address two difficult problems in the field of image segmentation:

1. Relating topological differences between two segmentations to an underlying pixel or voxel representation (Chapter 7).

2. Machine learning of boundary detection in a manner that focuses the classifier on topological rather than geometric accuracy (Chapter 8).

There is a history of interaction between digital topology and computer vision; indeed, substantial theoretical progress has been motivated by the practical importance of digital image processing. This interaction has largely occurred within the medical image processing realm, in which topology is used as a global constraint in non-learning based segmentation techniques [95].

In this chapter, we first present basic concepts in the field of digital topology as a brief review of prior work. We then use these concepts to introduce a novel classification scheme that is able to characterize the precise nature of a topological change caused by flipping a single pixel of a digital image. In Chapter 8 we show how this classification scheme can provide even more flexibility to the general topological learning scheme we later introduce.

6.1 Fundamental concepts

Topology is a major and unifying branch of mathematics that studies properties of objects that are invariant under certain kinds of transformations. Informally, the goal of a topological characterization is to abstract away from the geometry of objects to characterize higher order properties that are preserved under transformation. Various levels of topological equivalence can be defined depending on the chosen set of continuous transformations. Two major concepts are *homeomorphism* and *homotopy*.

Definition 6.1. Homeomorphism

A function $f : X \to Y$ from a topological space X into a topological space Y is called a homeomorphism if it is a continuous, one-to-one transformation with a continuous inverse f^{-1}.

Note that homeomorphism is a relation between two topological *spaces*. This is contrast to homotopy, which is defined as a relation over two *functions*:

Definition 6.2. Homotopy

A homotopy between two continuous functions $f, g : X \to Y$ is defined to be a continuous function $H\colon X \times [0,1] \to Y$ such that, for all points $x \in X$, $H(x,0) = f(x)$ and $H(x,1) = g(x)$. If two maps f and g have some homotopy connecting them, then f and g are homotopic (or $f \simeq g$).

Given two topological spaces X and Y, we say they are homotopy equivalent or of the same homotopy type if there exist continuous maps $f : X \to Y$and $g : X \to Y$ such that $g \circ f$ is homotopic to the identity map on X and $f \circ g$ is homotopic to the identity map on Y. Homotopy is a more general notion than homeomorphism. For example, a solid ball is homotopy equivalent to a single point, even though they are not homeomorphic[1]. Hence the precise meaning and implications of "topological equivalence" depends on whether two spaces X and Y are required to share the same homotopy type, or be homeomorphic, isotopic, etc. We will be explicit about which topological properties are preserved when we

[1]Spaces that are homotopy equivalent to a single point are called *contractible*.

refer to a procedure as "topologically preserving." In general, however, we will be considering homotopically preserving transformations which will allow, for example, objects to be contracted and thus lacking the crucial one-to-one bijective property of homeomorphisms.

6.1.1 Digital images

In this thesis we are interested in digital topology, which develops the concepts of continuous-space topology to the case of digital images. This is not trivial, as care has to be taken to establish definitions in the digital space such that meaningful and consistent notions of topological space and topologically-preserving deformation can be defined.

A digital image exists in the digital *plane*, which is the set of lattice points in Euclidean n-dimensional space. $\mathbb{Z}$ is the set of integers and thus $\mathbb{Z}^2$ is the 2d digital plane and $\mathbb{Z}^3$ is the 3d digital plane. An element of $\mathbb{Z}^2$ is a point $\mathbf{p} = (x, y) \in \mathbb{Z}^2$ where x and y are both integers that are the *coordinates* of $\mathbf{p}$. Similarly an element of $\mathbb{Z}^3$ is a point $\mathbf{p} = (x, y, z) \in \mathbb{Z}^3$.

A digital image contains a finite set $W \subset \mathbb{Z}^n$ and its infinite complement $\overline{W} = \mathbb{Z}^n \backslash W$. Elements of W are points $p \in Z^n$, which are collectively referred to as the foreground of the image. The infinite set $\overline{W}$ is the background of the image.

We thus define an n-dimensional binary image as a quadruple $(\mathbb{Z}^n, \kappa, \overline{\kappa}, W)$ where κ and $\overline{\kappa}$ are respectively adjacency relations used for W and $\overline{W}$. We discuss adjacency relations in the following section. However, we note that another interpretation of the digital plane that will be useful is to interpret the coordinates of $\mathbb{Z}^n$ as the vertices of an n-dimensional graph, where the edges between vertices are given by the neighborhood relationship between points.

6.1.2 Adjacency, connectivity, and components

We are now interested in establishing a notion of an object in the plane that will be a collection of foreground or background points. In order to group points together, we require some notion of connectivity between points that will specify the conditions under which two points will be considered connected. In digital images, it is natural to consider translation-invariant definitions based on proximity within the digital plane, also known as *adjacency*

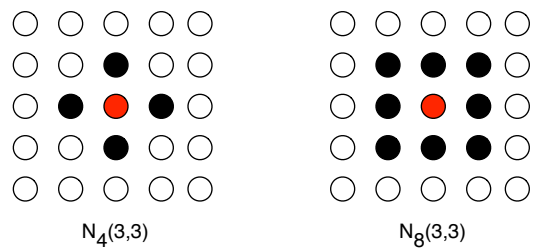


Figure 6-1: Adjacency in 2d. Open circles are background, and filled circles are foreground. The black filled circles are in the neighborhood of the red filled circle at $(3, 3)$, depending on whether 4-adjacency or 8-adjacency is used.

criteria. We define the neighborhood $N_k(\mathbf{p})$ of a point $\mathbf{p}$ as the set of all points that are k-adjacent to $\mathbf{p}$.

In the 2d plane, two points are 4-adjacent if their coordinates differ by at most 1 summed over all directions ($N_4(i, j) = \{(i \pm 1, j), (i, j \pm 1)\}$) and 8-adjacent if their coordinate differ by at most 1 in each direction: ($N_8(i, j) = N_4(i, j) \cup \{(i \pm 1, j \pm 1)\}$). These are illustrated in Figure 6-1.

In 3d images, we are generally interested in 6-adjacency ($N_6(i, j, k) = \{(i \pm 1, j, k), (i, j \pm 1, k), (i, j, k \pm 1)\}$), 18-adjacency ($N_{18}(i, j, k) = N_6(i, j, k) \cup \{(i, j \pm 1, k \pm 1), (i \pm 1, j, k \pm 1), (i \pm 1, j \pm 1, k)\}$) and 26-adjacency ($N_{26}(i, j, k) = N_{18}(i, j, k) \cup \{(i \pm 1, j \pm 1, k \pm 1)\}$).

In practice, two adjacency relationships are used in a digital image: one for the foreground (κ), and another for the background ($\overline{\kappa}$). The reasons for this are discussed in the next section; a typical choice, however, is $(\kappa, \overline{\kappa})$=(4,8) in 2d or $(\kappa, \overline{\kappa})$=(6,26) in 3d.

A κ-path γ is a sequence of points $x_0, x_1, ..., x_k$ where x_i is κ-adjacent to x_{i+1} for $i = 0, 1, ..., k - 1$. The path is *closed* if $x_0 = x_k$.

Having established the digital image space, points within the space, and a notion of connectivity specified by adjacency relationships, we define a set $S \subset \mathbb{Z}^n$ as κ-connected if S cannot be partitioned into two subset are not κ-adjacent to each other. A κ-component of a set S is a non-empty subset of S that is not κ-adjacent to any other point in S. Equivalently, an object S is κ-connected if for any two points $x, y \in S$, there is a κ-path γ, $\gamma \subset S$, from x to y.

A digital *curve* C is a finite, non-empty, κ-connected subset of $\mathbb{Z}^n$ in which each element has exactly two neighbors in C [88, 15]. Then we say that two digital curves C and D

differ by a *local deformation* if every element of C coincides with or is a neighbor of an element in D, and vice-versa. These definitions can be used to develop a rigorous definition of digital homotopy as a sequence of local deformations with certain constraints, as developed in [14, 15, 88].

An object is *simply connected* if every possible path $\gamma, \gamma \subset S$, can be reduced by a sequence of local deformations to a single point. There is a *hole* in S if there is a closed path contained in S that cannot be locally deformed in S to a single point (such paths can exist in either component of W (κ-paths) or any components of $\overline{W}$, excluding the infinite background component).

The *objects* in W are the set of κ-connected components of W. The *background* is the unique infinite $\overline{\kappa}$-connected component of $\overline{W}$. Any other $\overline{\kappa}$-connected component of $\overline{W}$ is called a cavity in W. In both 2d and 3d cavities are always regions in the sense of a bounded set of points. Moreover, in 2d a hole *is* a cavity and thus some well-defined set of points. However in 3d the situation for holes *(*sometimes called *tunnels* or *handles*) is more subtle. For example: a solid torus has one hole (any closed κ-path that loops around the interior of the torus) and zero cavities. Thus the hole itself is the missing "patch" within the torus, even though it is detect by a path in the interior of the torus. A hollow torus has one cavity and two holes (one given by closed κ-path along the surface of torus that loops around the entire object, and another formed by closed paths that form a loop around the interior cavity), and a hollow ball has zero holes and a single cavity.

6.1.3 Complementary adjacency and Jordan curves

In Euclidean topology, the Jordan curve theorem states that every non self-intersecting loop (also called a Jordan curve) in the plane divides the plane into an "inside" and "outside" region, and any path drawn from a point in one region to a point in the other region must intersect the curve at some point. This statement seems fairly obvious in the case of continuous space, but is non-trivial to prove, especially for curves that are nowhere differentiable such as the Koch snowflake.

We would like a concept of a Jordan curve in digital space for which a "digital" Jordan curve theorem can be stated. To understand the importance of this guarantee, imagine

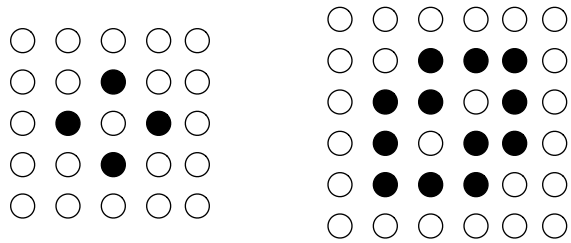


Figure 6-2: Illustration of two kinds of connectivity paradox in 2d when $\kappa = \overline{\kappa} = 8$ for the image on the left, and $\kappa = \overline{\kappa} = 4$ for the image on the right.

if it did not exist: some closed curves would have a connected path from the background component into the interior of an object. Hence the notion of a boundary would be ill-defined. The ability to formulate a digital Jordan curve theorem depends crucially on the choice of adjacency used to define connectivity, and will thus impose restrictions on the choice of adjacency we are able to use in practice.

To illustrate these issues, consider the 2d digital images in Figure 6-2. Open circles correspond to background points and filled circles correspond to foreground points. For the left figure, if 8-adjacency is used for both foreground and background points ($\kappa = \overline{\kappa} = 8$), then there is a single foreground component *and* a single background component because the middle background point is 8-adjacent to the rest of the background. Hence the 8-adjacent non self-intersecting curve in foreground space fails to divide the plane into an inside and outside region. For the figure on the right, assume that 4-adjacency is used for both foreground and background points ($\kappa = \overline{\kappa} = 4$); there will be one foreground component and *three* background components, because the two interior background points are not adjacent to each other. Thus the Jordan curve criteria are again violated, as a single non-intersecting curve has divided the plane into *two* disconnected inside regions.

Fortunately there is a simple resolution to this problem, and that is to use distinct adjacency criteria for foreground and background components. In 2d, we require that $(\kappa, \overline{\kappa}) = (4, 8)$ or $(\kappa, \overline{\kappa}) = (8, 4)$; we refer to these compatible pairs as *complementary* adjacencies. For such adjacencies, it is possible to state the 2d Jordan curve theorem:

Theorem 6.3. *2d Digital Jordan Curve Theorem. Given a κ-curve $C \subset \mathbb{Z}^2$ with at least four points if $\kappa = 8$ and at least eight points if $\kappa = 4$, then $Z^2 \backslash C$ consists of two $\overline{\kappa}$-connected sets. Exactly one of these sets is finite and is called the interior with respect to C and the*

other is infinite and is called the exterior with respect to C.

Proofs of this theorem and similar theorems for compatible adjacencies in 3d such as $(\kappa, \overline{\kappa}) = (6, 26)$ can be found in [50, 52, 51].

6.1.4 Topology preservation and simple points

A major practical goal of digital topology is to identify methods of altering the *geometry* of objects within a digital image without altering any topological properties of the image. Of course, this requires us to specify what we precisely mean by the topology of a digital image. In 2d, a highly useful definition of topological equivalence is given by considering whether two images have isomorphic *adjacency trees* [89]. For some image I, an adjacency tree is constructed with the infinite background component B at its root and then considering the nested and alternating relationship of finite foreground objects and holes (with the leaves of the tree corresponding to components that have no holes). Note that this is a different notion of equivalence than simply considering those images which have an equivalent number of objects and holes to be topologically equivalent. In particular, one can consider examples of 2d images which have an equivalent number of holes and objects but cannot be continuously deformed from one to the other while preserving topology during the deformation (consider an image X with two concentric circles and an image Y with two independent circles; X and Y have the same number of holes (two) and objects (two), but difference adjacency trees).

The topology of 3d binary images is considerably more complicated than that of 2d spaces. Hence topological equivalence in 3d space has taken on various definitions in the literature, with most authors simply committing to the number of components, holes, and tunnels as the required equivalence between topologically equivalent 3d images [67].

We would now like to identify points in a digital image that can be modified (i.e., "flipped" from foreground to background or vice versa) without altering the numbers of components, cavities, and in the case of 3d images, tunnels. More precisely, given a set W' obtained from W by flipping p, a point p is *κ-simple* (or more commonly referred to as a simple point) if and only if:

- W' has the same number of κ-connected components (objects in W').

- $\overline{W}'$ has the same number of $\overline{\kappa}$-connected components (cavities in W').
- W' has the same number of handles (only applicable to 3d images).

Thus by definition flipping a "simple" point is a topology-preserving operation and flipping a "non-simple" point is not topology preserving. For 2d topology, it has been proven that not only are altering simple points topology-preserving, but that the converse is true: two images that are topologically equivalent in the sense of sharing isomorphic adjacency trees can always be transformed into each other by a sequence of changes in the values of simple pixels [89].

Although flipping a single simple point is guaranteed to preserve topology, it is *not* true that simultaneously flipping and arbitrary set of *multiple* simple points will preserve topology. Hence, many algorithms that deform digital images by altering simple points instead perform a sequence of flips, where any particular flip is made at a point that is simple relative to the current image in the sequence. Since all flips preserve topology, such a sequence of flips is a topology-preserving deformation of the original image (sometimes called a homotopic deformation [95]). There has also been some work on explicitly identifying sets of points which can be flipped simultaneously without altering topology [78, 79].

In practice, an issue of considerable importance is the efficient characterization of a particular point within a digital image as being simple or non-simple. Fortunately, this can be done locally for both 2d and 3d images: a point $\mathbf{p}$ is simple if and only if flipping this point does not change the number of foreground components in $N_\kappa(\mathbf{p})$ and the number of background components in $N_{\overline{\kappa}}(\mathbf{p})$. Several other even more efficient methods of identifying 2d and 3d simple points have been discovered [52, 8].

Bertrand has introduced a method for identifying simple points that relies on the very useful concept of *topological numbers* [7, 8]. The topological number $T_\kappa(\mathbf{p}, W)$ is the number of foreground connected components in the neighborhood of p (i.e., the 3×3 patch centered at p in 2d, and the $3 \times 3 \times 3$ patch centered at p in 3d) under κ-adjacency, and the topological number $T_{\overline{\kappa}}(\mathbf{p}, \overline{W})$ is the number of background connected components in the neighborhood of a point p under $\overline{\kappa}$-adjacency. A point p is κ-simple if and only if $T_\kappa(\mathbf{p}, W) = T_{\overline{\kappa}}(\mathbf{p}, \overline{W}) = 1$, which is proved in [8].

6.2 Classification of non-simple points

Non-simple points are those elements of a binary image which, when flipped either from foreground to background or background to foreground, somehow alter the topology of the image. As established in Section 6.2, by "change topology" we mean alter the number of components, holes, or cavities. Although straightforward procedures exist to check whether a particular location is simple or not, it will also be useful to be able to characterize the *type* of topological change associated with a particular non-simple point. In other words, we would like to know which topological quantity is affected by flipping a particular non-simple point.

6.2.1 Identifying addition and deletion of objects using topological numbers

Certain cases can be identified based on topological number alone (see Section 6.1.4 for a discussion of topological numbers). Let $(\kappa, \overline{\kappa})$ be some complementary adjacency relation in either a 2d or 3d space.

Theorem 6.4. *Topological number characterization of object deletion. Suppose q is an element of the foreground W. Then flipping q will result in a κ-component deletion if and only if $T_\kappa(q, W) = 0$.*

Proof. The proof (given in [8]) is simple: suppose that q is flipped, then a connected component of W is removed if and only if q is an isolated point, in which case $T_\kappa(q, W)$=0. Conversely suppose $T_\kappa(q, W) = 0$, then q must be an isolated point with respect to the foreground. Therefore flipping q results in an object deletion. □

Note that object deletion is only one way to decrease the number of connected components. The merging of two existing components is the other way; we will discuss this case shortly.

A similar statement can be offered for object addition:

Theorem 6.5. *Topological number characterization of object addition. Suppose q is an*

element of the background $\overline{W}$. Then flipping q will result in a κ-component addition if and only if $T_\kappa(q, W) = 0$.

Proof. Suppose that q is flipped, then a connected component W is added if and only if q is an isolated point, in which case $T_\kappa(q, W) = 0$. Conversely suppose $T_\kappa(q, W) = 0$, then q must be an isolated point with respect to the foreground. Therefore flipping q results in an object addition. □

Again, we note that object addition is only one way to add to the number of components. Splitting an existing object into two is the other.

Topological numbers can also be used to identify the creation and deletion of cavities, using a similar criteria based on the "background" topological number. Recall that the background is the unique infinite $\overline{\kappa}$-connected component of $\overline{W}$, and by definition any other $\overline{\kappa}$-connected component of $\overline{W}$ is called a cavity in W.

Theorem 6.6. *Topological number characterization of cavity deletion. Suppose q is an element of the background $\overline{W}$. Then flipping q will result in a $\overline{\kappa}$-component (cavity) deletion if and only if $T_{\overline{\kappa}}(q, \overline{W}) = 0$.*

Proof. Suppose that q is flipped, then a connected component of $\overline{W}$ is removed if and only if q is an isolated background point, in which case $T_{\overline{\kappa}}(q, \overline{W})$=0. Conversely suppose $T_{\overline{\kappa}}(q, \overline{W}) = 0$, then q must be an isolated background point. Therefore flipping q results in deletion of a cavity. □

Theorem 6.7. *Topological number characterization of cavity addition. Suppose q is an element of the foreground W. Then flipping q will result in a $\overline{\kappa}$-component (cavity) addition if and only if $T_{\overline{\kappa}}(q, \overline{W}) = 0$.*

Proof. Suppose that q is flipped, then a connected component in $\overline{W}$ is added if and only if q is an isolated background point, in which case $T_{\overline{\kappa}}(q, \overline{W}) = 0$. Conversely suppose $T_{\overline{\kappa}}(q, \overline{W}) = 0$, then q must be an isolated point with respect to the background. Therefore flipping q results in the creation of cavity. □

As in the case of foreground components, deletion/addition of cavities by flipping isolated background points is *not* the only way in which the number of background components can be

changed. A cavity can be merged (with other cavities or the infinite background component) and split into two cavities by a single flip of some pixel q. However in such cases it will not be true that $T_{\overline{\kappa}}(q, \overline{W}) = 0$.

We note that if $T_\kappa(q, W) = 0$ then it follows $T_{\overline{\kappa}}(q, \overline{W}) = 1$ and vice versa. This is because an isolated foreground (background) point is clearly surrounded by a background (foreground), which under normal adjacency relations for either κ or $\overline{\kappa}$ will be connected as a single component within such a neighborhood itself. Finally we recall that if $T_\kappa(q, W) = T_{\overline{\kappa}}(q, \overline{W}) = 1$ the point is simple and thus causes no topological change when altered. Therefore, we have a characterization for all points for which $T_\kappa(q, W) \in \{0, 1\}$ and $T_{\overline{\kappa}}(q, \overline{W}) \in \{0, 1\}$.

6.2.2 Identifying splits, mergers, and hole addition/deletion using extended topological numbers

Topological numbers as originally defined are insufficient to characterize the precise nature of topological changes caused by flipping non-simple points for which $T_{\overline{\kappa}}(q, \overline{W}) > 1$ or $T_\kappa(q, W) > 1$. For example, flipping a pixel q from background to foreground when $T_\kappa(q, W) > 1$ clearly implies a topological change since q is non-simple; however, the nature of this change depends on *non-local* properties that are not captured by the local neighborhood from which $T_\kappa(q, W)$ is computed. We need access to global connectivity information that would enable us to distinguish between, for example, merging two objects versus creating a hole. Global connectivity is well defined over the image, as discussed in Section 6.1 and Section 6.2, however the binary image itself does not represent such information directly. Hence, we will need to consider additional measures beyond topological numbers that take into account this global information.

A useful concept for this purpose introduced by Segonne [94] is the *extended* topological numbers of some point q: $T^+_\kappa(q, W)$ and $T^+_{\overline{\kappa}}(q, \overline{W})$. We define $C_\kappa(q, W)$ to be the set of κ-connected components of $W \backslash \{q\}$ that are κ-adjacent to q. Then $T^+_\kappa(q, W) = |C_\kappa(q, W)|$ and $T^+_{\overline{\kappa}}(q, \overline{W}) = |C_{\overline{\kappa}}(q, \overline{W})|$. In other words, these quantities count the number of unique foreground and background connected components in the local neighborhood of q, but with

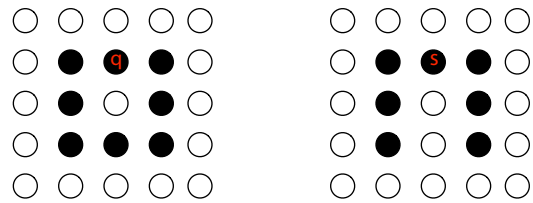


Figure 6-3: Under $(\kappa, \overline{\kappa}) = (4, 8)$ adjacency, flipping the foreground pixel q in the left image, for which $T_\kappa(q, W) = 2$ and $T_k^+(q, W) = 1$, results in the deletion of a cavity. Flipping the the foreground pixel s in the right image, for which $T_\kappa(q, W) = 2$ and $T_k^+(q, W) = 2$, results in a split.

global connectivity taken into account. This is a very useful quantity because it allows us to measure the global topological effect of a local change in the image. In Figure 6-3 we illustrate the difference between T_κ and T_κ^+ in a simple 2d example.

6.2.2.1 Mergers and splits

We will refer to any flip that results in a reduction in the number of connected components in W resulting from the addition of a foreground point as a *merger*. We will refer to any increase in the number of connected components of W resulting from the deletion of a foreground point as a *split*.

Theorem 6.8. *Extended topological number characterization of mergers. Suppose q is an element of the background $\overline{W}$. Then flipping q will result in a reduction in the number of total foreground components if and only if $T_\kappa^+(q, W) > 1$.*

Proof. Suppose that q is flipped to the foreground. Then the number of connected is reduced only if a path is created by q between two or more unique components in W, in which case $T_\kappa^+(q, W) > 1$. Conversely suppose $T_\kappa^+(q, W) > 1$, then q must be adjacent to more than one unique connected component. Therefore flipping q to foreground results in the creation of a path between the unique components κ-adjacent to q. Therefore the total number of components is reduced (by exactly $T_\kappa^+(q, W) - 1$). □

Theorem 6.9. *Extended topological number characterization of splits. Suppose q is an element of the foreground W. Then flipping q will result in an increase in the number of total foreground components if and only if $T_\kappa^+(q, W) > 1$.*

Proof. Recall that $T_\kappa^+(q, W)$ gives us the number of unique connected components adjacent to q in $W \backslash \{q\}$. Suppose that q is flipped to the background, then the number of connected components will increase only if a path between two or more unique components in W was provided by q, in which case $T_\kappa^+(q, W) > 1$ (if there was any other κ-connected path between two components adjacent to q, then they would not be distinct components in $C_\kappa(q, W)$). Conversely suppose $T_\kappa^+(q, W) > 1$, then q must be adjacent to more than one unique connected component. Therefore flipping q to background results in the destruction of a path between any unique components κ-adjacent to q. Therefore the total number of components is increased (by exactly $T_\kappa^+(q, W) - 1$). □

Similar statements can be made about cavities (background components). In particular, if q is an element of the foreground W, then flipping q will result in a reduction in the number of total background components if and only if $T_{\overline{\kappa}}^+(q, \overline{W}) > 1$. If q is an element of the background $\overline{W}$, then flipping q will result in an increase in the number of total background components if and only if $T_{\overline{\kappa}}^+(q, \overline{W}) > 1$. We neglect the proofs as they are similar as to those of the foreground component case.

6.2.2.2 Holes

Thus far the discussion of non-simple point characterization has been general to either the 2d or 3d case. Now, however, we turn our attention to holes (also known as tunnels or handles). In 3d, deleting or adding a hole does not necessarily result in a change in the number of foreground or background components. Note that a single flip can be associated with a merger or split *and* a change in the number of holes due to the interaction of multiple components through a single point.

First, we note a basic relationship between standard and extended topological numbers: $T_\kappa^+(q, W) \leq T_\kappa(q, W)$ and $T_{\overline{\kappa}}^+(q, \overline{W}) \leq T_{\overline{\kappa}}(q, \overline{W})$. This relationship is true because the global image can only *add* and not destroy paths between κ or $\overline{\kappa}$ adjacent elements. Therefore the number of unique components based on global connectivity will always be less or the same as the number based on just local connectivity [94].

Second, we note that the existence of a tunnel is characterized by a closed path within

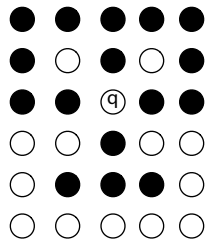


Figure 6-4: Under $(\kappa, \overline{\kappa}) = (4, 8)$ adjacency, flipping q from background to foreground results in a merger and the addition of two holes (2d cavities). $T_\kappa(q, W) = 4$, $T_\kappa^+(q, W) = 2$, $T_{\overline{\kappa}}(q, \overline{W}) = 4$, and $T_{\overline{\kappa}}(q, \overline{W}) = 3$.

an object that cannot be deformed to a single point. Hence, the addition of a handle can be detected by the addition of a foreground point which adds a path from an object to itself in a non-simple location (otherwise it would simply be a border point). Removal of such a point constitutes the deletion of a handle. In other words, for some single component C, the condition $T_\kappa(q, C) > 1$ implies that the addition of point q generates at least one or more handles [94].

Theorem 6.10. *Extended topological number characterization of κ-hole addition. Suppose q is an element of the background $\overline{W}$. Then flipping q will result in an increase in the number of κ-holes if and only if $1 \leq T_\kappa^+(q, W) < T_\kappa(q, W)$.*

Proof. Suppose we flip q to the foreground. Then the number of κ holes can be increased only if a new closed path is added within some κ-component that is κ-adjacent to q. A closed path between κ-adjacent points of q could only be added by q if those points already belong to the same component but are not locally connected in the absence of q, implying that $1 \leq T_\kappa^+(q, W) < T_\kappa(q, W)$. Conversely if $1 \leq T_\kappa^+(q, W) < T_\kappa(q, W)$ then there exists at least two κ-adjacent elements of q which are not κ-adjacent to each other, but part of the same global component (are connected by some κ-path). Hence flipping q creates a new closed path between in this component and thus a κ-hole is added. □

Theorem 6.11. *Extended topological number characterization of κ-hole deletion. Suppose q is an element of the foreground W. Then flipping q will result in a decrease in the number of κ-holes if and only if $1 \leq T_\kappa^+(q, W) < T_\kappa(q, W)$.*

Proof. Suppose we flip q to the background. Then the number of κ holes can be decreased

$T_\kappa(q,W)$	$T_{\overline{\kappa}}(q,\overline{W})$	$T_\kappa^+(q,W)$	$T_{\overline{\kappa}}^+(q,\overline{W})$	$q \in W$	$q \in \overline{W}$
0	1	0	1	Object deletion	Object addition
1	0	1	0	Cavity addition	Cavity deletion
1	1	1	1	No change	No change
> 1		> 1		Object split	Object merger
> 1	> 1		> 1	Cavity merger	Cavity split
$> T_\kappa^+(q,W)$		≥ 1		Hole deletion	Hole creation

Table 6.1: Topological change caused by flipping q.

only if an existing closed path is removed within some κ-component that is κ-adjacent to q. A closed path between κ-adjacent points of q could only be removed if those points already belong to the same component but are not locally connected in the absence of q, implying that $1 \leq T_\kappa^+(q,W) < T_\kappa(q,W)$. Conversely if $1 \leq T_\kappa^+(q,W) < T_\kappa(q,W)$ then there exists at least two κ-adjacent elements of q which are not κ-adjacent to each other, but part of the same global component (are connected by some κ-path). Hence flipping q removes an existing closed path within this component and thus a κ-hole is deleted. □

6.2.2.3 Summary of classification scheme

In table 6.1 we summarize the classification scheme of non-simple points. In the first three rows, both normal and extended topological numbers will always take on the same value. Note that certain topological changes in the same column can occur simultaneously if multiple row-conditions are satisfied (for example, flipping a single point q from background to foreground can merge two foreground objects and split a cavity).

6.2.3 Implementation issues

Calculating standard topological numbers, which depend on purely local information, is straightforward and can be done in constant time using boolean criteria [8].

De terming the extended topological number of an image location is more involved. This is because the extended topological numbers depend on non-local properties (potentially any path in the image). A reasonable method for dealing with this is proposed in [94]: a label map L is created that assigns each foreground and background component (according to the chosen $(\kappa, \overline{\kappa})$ adjacency) a unique label. This computation can be done efficiently using

standard connected components algorithms.

If $q \in \overline{W}$, then $T_\kappa^+(q, W)$ is directly determined by counting the number of unique foreground components in the neighborhood of q in L. However to compute $T_{\overline{\kappa}}^+(q, \overline{W})$, it is necessary to remove q from $\overline{W}$ (i.e., flip q from background to foreground) and then recompute the connected components of any background components adjacent to q. $T_{\overline{\kappa}}^+(q, \overline{W})$ is then given by the number of unique background components in the neighborhood of q in this re-computed label map.

If $q \in W$, then $T_{\overline{\kappa}}^+(q, \overline{W})$ is directly determined by counting the number of unique background components in the neighborhood of q in L. However to compute $T_\kappa^+(q, W)$, it is necessary to remove q from W (i.e., flip q from foreground to background) and then recompute the connected components of any foreground components adjacent to q. $T_\kappa^+(q, W)$ is then given by the number of unique foreground components in the neighborhood of q in this re-computed label map.

Hence, component mergers can be classified directly from the label map L. Intuitively, this is because a merger involves adding a path between distinct components (a relationship directly reflected in L). A split is computationally more difficult to detect because there may be paths that non-locally connect some locally adjacent image locations. If, however, it is known that the component associated with some *non-simple* foreground location q has no handles, then flipping q from foreground to background *must* result in a split if $T_\kappa(q, W) > 1$.

6.3 Segmentation by connected components

A segmentation is a partition of a pixel or voxel locations into a set of non-overlapping regions that cover the image space. We can regard a segmentation as an assignment over N pixels $X = \{x_1, ..., x_N\}$ into a discrete set, where a particular assignment for pixel location i is given by $l_i^S \in [1, k]$ for a segmentation S with k separate objects. In this abstract formulation, a segmentation can be considered a clustering of pixel locations, a fact which has motivated prior work in using the Rand index as an image segmentation metric [105]. Metrics will be discussed in much greater detail in the next chapter.

Generally speaking, a segmentation is considered correct when the regions are consistent

Figure 6-5: Schematic of segmentation based on connected components. An analog-valued image (left) is converted by some function into a *binary*-valued image (middle) that is associated with a particular $(\kappa, \overline{\kappa})$ adjacency. The adjacency and binary image is then used to generate a segmentation (right) under the definition of a connected component given in Section 6.1.2.

with the physical identity of the objects that are being imaged. From the computational point of view, this task is conceptually very different from other vision problems such as categorization or classification; instead of assigning an image some global identity, we are producing another image which represents a grouping of image pixels.

Digital topology provides a framework for image segmentation that is potentially very powerful because it defines a well-defined link between the image space, a segmentation, and topology. The overall approach is illustrated in Figure 6-5. An analog-valued image is converted by some function into a *binary*-valued image that is associated with a particular $(\kappa, \overline{\kappa})$ adjacency. The adjacency and binary image is then used to generate a segmentation under the definition of a connected component given in Section 6.1.2. The coloring of the object components is arbitrary and is simply a means by which to represent the non-local concept of connectedness.

The binary image generated as an intermediate result in this process is a digital image sometimes referred to as an "in/out" labeling, due to the partitioning of the image space into either foreground "inside object" pixels or background "outside object" pixels. An in/out labeling is nearly equivalent to another concept called a "boundary labeling," in which a pixel is assigned to foreground if it belongs to a boundary between objects, and is background otherwise [70]. A minor difference is that in a boundary labeling the background is treated as an object and in an in/out labeling the background is not. Except for this difference, flipping the binary values of one representation will yield the other. The relationship between these representations for a natural image that was segmented by a human is illustrated in Figure

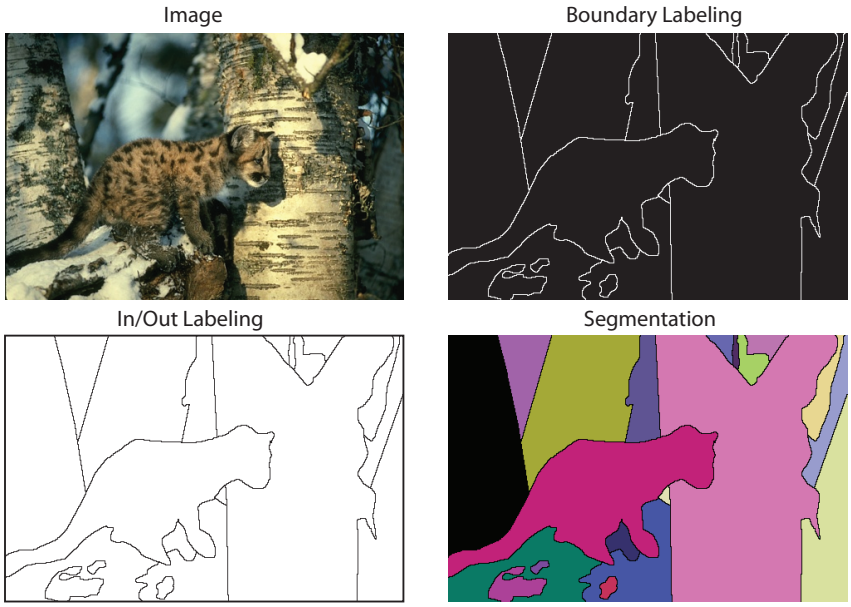


Figure 6-6: In/out and boundary labeling for a hand-traced natural image; white pixels denote a binary value of 1 and black pixels are 0. The segmentation is given by connected components of the in/out image with 4-adjacency.

6-6.

6.3.1 Affinity graphs

By the definition of a connected component, distinct objects cannot have foreground pixels κ-adjacent to each other in their corresponding representation as a digital image (they would be considered the same component since there is a κ-path between them). In other words, under a typical inside/outside representation of components in a binary image there must always be some background component separating any two distinct foreground components. In certain situations, this may prove to be a limitation. For example, if the resolution of the image is lower than the physical size of a boundary, then the most natural representation of a segmentation may be one in which there is no outside space separating objects and simply a transition from one object to the next.

A solution to this problem is to consider digital images defined under *edge* topology rather than pixel or voxel topology. In this idea, each pixel or voxel is a node in a graph, and edges between neighboring nodes are mapped to a binary value. An adjacency relation

among neighboring edges (sometimes called affinities) can be defined such that components are formed by paths of connected edges. This type of representation is sometimes called an "affinity graph." This strategy overcomes the out-space requirement of binary image representations and has been successfully employed in prior segmentation work, including the electron microscopy segmentation problem discussed in this thesis [30, 104]. The topology of images defined over edge-space is more subtle than that of the topology of pixel-space, but can indeed be defined along with concepts such as a simple edge [29, 12].

Chapter 7

Warping Error: A Novel Segmentation Metric

Abstract. We show how concepts from digital topology can be used to define a novel metric for the quantitative evaluation of the performance of a segmentation algorithm. We first discuss general challenges related to the evaluation of segmentation accuracy and notions of ground-truth in segmentation, using examples from both natural image and electron microscopy datasets. We then discuss properties of our warping-based error metric, compare it to existing approaches, and show how it can be used to characterize segmentation errors.

Quantitative metrics are obviously crucial for comparing one algorithm against another; within the machine learning context, metrics can also be used to guide the optimization of an algorithm. For those researchers interested in pursuing a machine learning approach to image segmentation, an ideal metric would therefore both accurately assess performance *and* be amenable to some kind of efficient optimization during supervised learning. In the following chapter, we describe a method to optimize the metric we introduce.

7.1 Introduction

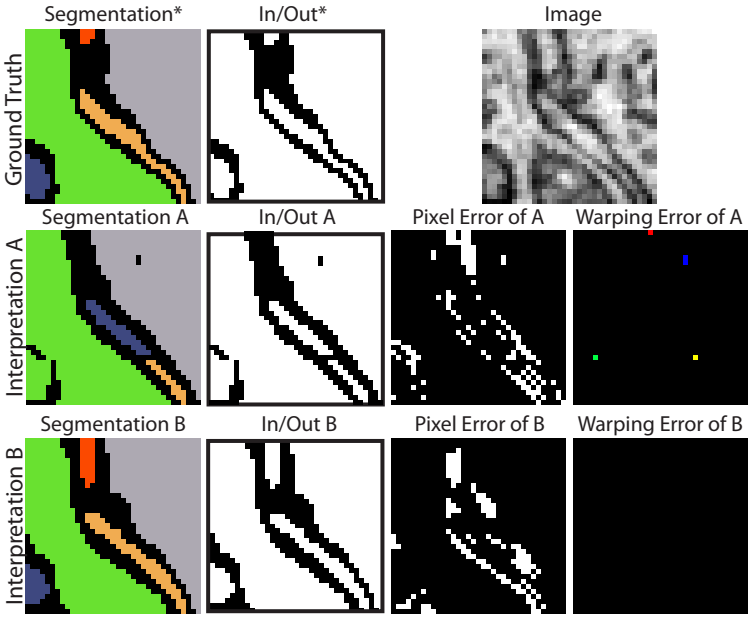
Many segmentation methods in the literature have been evaluated on a largely subjective and qualitative basis, with visual inspection the primary means of comparing techniques to one another. Since the introduction of hand labeled ground-truth datasets for segmentation,

however, there has been increased interest in methods for quantifying accuracy [68, 71, 70, 105].

In high-level image analysis tasks such as object recognition, an appropriate metric is usually straightforward to define because the output space is typically low-dimensional with minimal ambiguity regarding correct answer for any output variable. Image segmentation presents a more challenging scenario for evaluation. Fundamentally, this is because segmentation is a structured prediction problem in which interpretation of a single input (an image) typically involves the prediction of a high-dimensional set of variables (for example, boundary locations) for which there is no uniquely correct configuration. Instead, it is more appropriate to think of the ground truth as providing *constraints* on mutually dependent and correlated output variables. In this setting, the goal of the classifier is to produce a configuration of output variables which, for example, minimizes the amount of violation of the constraints.

One line of evidence supporting this view of the segmentation problem comes from examining inter-annotator discrepancy in human segmentations of natural images or electron microscopy images. In either case, humans tend to disagree on the exact shape of objects and placement of boundaries. Hence, a simple comparison of the segmentations by measuring the consistency of boundary placement in terms of "pixel error" (defined shortly) yields surprisingly low agreement between humans; for example, a roughly 10% disagreement rate in the case of electron microscopy images. The source of this variability may include subjective differences in interpretation, errors associated with the acquisition of human annotation (e.g., hand jitter), and fundamental ambiguity.

However, even if boundary localization was not fundamentally ambiguous and it was possible to eliminate human variability or obtain true ground truth through other means, minor errors in boundary localization within an automated segmentation may *still* be considered insignificant compared to topological errors such as the merging and splitting of objects. We would like a metric that appropriately reflects the disparity in the significance of different types of errors.



	Error of A	Error of B	$\frac{ErrorB}{ErrorA}$
Pixel Error	115/1225	115/1225	1
Berkeley: 1-(F measure)	0.018	0.066	3.67
1-(Rand Index)	0.08	0.06	.75
Warping Error	5/1225	0/1225	0

Figure 7-1: Comparison of different metrics on a 35×35 pixel image from the training set used in Section 5. Within each segmentation, pixels with the same color correspond to a single object. Interpretation A has both topological and geometric differences with the ground truth, whereas Interpretation B has geometric discrepancies but no topological errors. The Rand Index and warping error show a large relative difference between interpretations B and A, while the pixel error does not. The Berkeley F-measure *favors* the topologically incorrect interpretation. Warping error of interpretation A shows four topological errors: red pixel denotes object deletion, green pixel denotes an object merger, yellow pixel denotes an object split, and a blue pixels denote creation of a hole. The Berkeley F-Measure and Rand index were converted to error measures by subtracting from 1, the value of perfect agreement for those measures. For evaluation of the Berkeley F-measure, a boundary labeling was computed by completely thinning a binary flip of the in/out map (in accordance with procedures defined in the benchmark code provided with [70]).

7.2 Existing metrics

In the literature of both low-level algorithms and evaluation metrics, a fundamental distinction is that between a boundary map and a segmentation. As discussed in Chapter 6, a boundary detector produces a $[0, 1]$-valued analog boundary map which can be interpreted as giving the probability at each image location of there being a boundary.

A boundary map is not a segmentation, because it does not itself provide a grouping of pixels into regions. However, it is traditional to train a classifier to produce a boundary map from which a segmentation is computed (using, for example, connected components on a thresholded boundary map or some other graph partitioning algorithm). Hence, one approach has been to quantify the accuracy of a boundary detector given ground-truth boundary maps produced *from* a ground truth segmentation. This is the goal of the pixel error and Berkeley segmentation benchmark, which we review in this section. We then discuss the Rand index, which is a metric that operates on true segmentations. In Figure 7-1 we compare the metrics, including the warping error introduced in Section 7.3, to one another for a sample image.

7.2.1 Pixel error

The simplest method of quantifying boundary map accuracy is what we refer to as "pixel error." Let $F_I(\vec{w})$ be the probability map produced by applying the classifier $f(\vec{w})$ to the entire image I. We can binarize the classifier output at some value θ and then compute the fraction of boundary locations that were incorrectly classified: $\frac{1}{|L|}\sum_i [l_i \neq (f_i(\vec{w}) > \theta)]$ where $|L|$ denotes the number of image locations. Precision and recall curves can then be computed by varying θ. Since the probability map $F_I(\vec{w})$ and a binary labeling L of the image I are both images of the same size, we can also compute their Euclidean distance $d(F_I(\vec{w}), L) = \sum_i [l_i - f_i(\vec{w})]^2$, where the labeling and probability map at location i are written as l_i and $f_i(\vec{w})$, respectively. This gives us a real valued error measure, which is often more suitable for optimization purposes.

The pixel error is a tempting metric because it is simple to compute, has an an intuitive interpretation, and its optimization is a well-studied problem in the machine learning liter-

ature. Unfortunately, it suffers from serious defects. Firstly, it is *overly* sensitive to minor displacements in the location of a boundary that are ubiquitous even when comparing one human boundary map to another. Qualitatively, these disagreements are ultimately minor differences in the interpretation of the image, but lead to large quantitative differences in pixel error. Secondly, pixel error is *insufficiently* sensitive to certain errors in a boundary map which are likely to lead to errors in a segmentation produced from the boundary map. Of course, the precise method used to generate a segmentation from a boundary map will determine which errors are the most relevant. However, it is reasonable to assume that, for example, a missing boundary between two directly adjacent objects will be more likely to cause a segmentation error than a geometric perturbation of boundary location.

A boundary map can have very high error but still be likely to produce a reasonable overall interpretation of an image in the scene of a segmentation. Conversely, a boundary map can have very low error but still be likely to produce an unreasonable overall interpretation of an image. Therefore, pixel error in its naive form is unsuitable as an error metric, especially if segmentation is the ultimate goal.

7.2.2 Berkeley segmentation benchmark

A boundary detection metric that improved on pixel error was introduced by Martin et al. [70]. In their approach, a correspondence between a machine boundary labeling and a human boundary labeling is computed by solving a minimum cost bipartite assignment problem, in which boundaries in one labeling are matched to boundaries in another labeling. In this optimization, the cost between two boundaries is proportional to their distance in the image plane. Their error measure considers a machine boundary pixel to be correct if it was within some distance cutoff of the corresponding human boundary pixel and thus tolerates small differences in boundary localization. This sophisticated error measure is currently used in the benchmark code provided with the Berkeley Segmentation Dataset.

To compute the error, the matching is converted from a dense assignment problem (with complexity between $O(n^2)$ and $O(n^3)$) to a sparse assignment problem that appears to have an empirical runtime of $O(n)$ [70, 22]. This process involves the creation of some random edges between nodes to ensure all nodes are matched. As a result, the metric does not

produce deterministic results (the randomness in the conversion to a sparse matching problem can have small effects on the output). Although the dense assignment problem is not NP-hard, the authors chose to employ a faster (near linear) approximate method as a practical measure. Hence, the result of this benchmark is not based on the true best matching, but on an approximate solution that in most cases is presumably close to the best solution.

In summary, the Berkeley segmentation benchmark operates directly on a boundary map and solves one of the major problems with pixel error. However, there are still serious limitations with this metric. First, the error measure is insufficiently sensitive to those errors in the boundary map that are likely to lead to segmentation errors. For example, these metrics are sometimes insensitive to gaps in boundaries, which can lead to mergers between two regions in a segmentation [3]. Second, it is not clear how to efficiently optimize this metric in the context of supervised machine learning.

7.2.3 Rand index

The Rand index is a well-known statistical measure of the similarity between two data clusterings [85]. Recently, the Rand index has also been proposed as a measurement of segmentation quality that can operate directly on a segmentation by interpreting a segmentation as a grouping of pixels into separate clusters [105]. More formally, consider an image space X with n elements $\{x_1, ..., x_n\}$ and two segmentations S and T which assign each pixel $\{x_i\}$ to an integer label $\{s_i\}$ and $\{t_i\}$ respectively. Then the Rand index is a $[0, 1]$-valued measure of the number of pairs of points having the same label relationship in S and T:

$$R(S,T) \quad = \quad \frac{1}{\binom{n}{2}} \sum_{i,j \neq i} \delta(S_i = S_j, T_i = T_j) + \delta(S_i \neq S_j, T_i \neq T_j)$$

The Rand index is 1 when the two segmentations perfectly match, and 0 when they completely disagree. Unnikrishnan et al. have also introduced the Normalized Probabilistic Rand (NPR) index to handle the situation of comparing to multiple ground truths [105].

The Rand index effectively focuses on whether the overall grouping of points into separate segments is correct; subtle geometric differences between two segmentations will decrease the Rand index but will be much less numerically significant than, for example, the erroneous

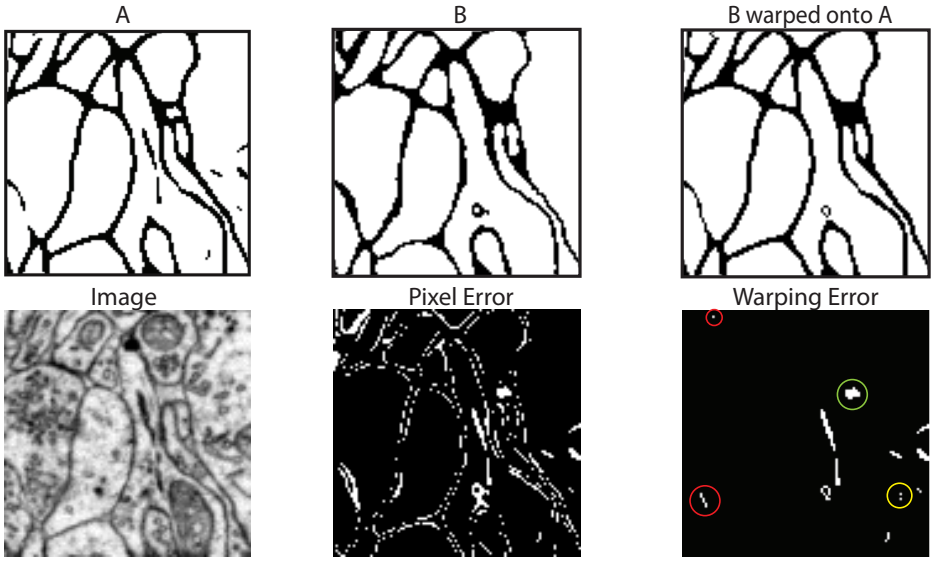


Figure 7-2: Pixel error compared to warping error for images taken from an electron microscopy dataset described in Chapter 8. Labeling B is warped onto A. In the classification scenario, A can be considered a candidate output and B the ground truth labeling. Pixel error is the difference mask of A and B; numerous geometric errors are present, in addition to topological errors. Warping error is the difference mask of A and "B warped onto A." Only topological errors are present: red circles in the warping error denote locations with merge errors, the green circle denotes object addition, the yellow circle denotes split errors, and all other differences correspond to hole (cavity) creation.

merging of two regions. Moreover, it has recently been discovered that it is possible to optimize the Rand index via gradient optimization [103], making the Rand index an attractive framework for both learning and evaluating image segmentation.

Conceptually the Rand index focuses on whether a segmentation has achieved the correct *connectivity* between pixels (i.e., whether two pixels are connected or disconnected correctly). This is in contrast to the warping-based topological error metric we introduce in the next section, which focuses on *topology*.

7.3 Warping-based error metric

Here we propose an error measure that tolerates boundary localization errors, much as in the Berkeley segmentation benchmark [70]. However, our measure has two additional virtues. First, it retains sensitivity to topological errors while tolerating boundary localization errors.

Second, it can be used for both training and testing image labeling algorithms.

Suppose we are given two binary images, $L^* = (\mathbb{Z}^n, \kappa, \bar{\kappa}, W_{L^*})$ and $T = (\mathbb{Z}^n, \kappa, \bar{\kappa}, W_T)$. If L^* can be transformed into L by a sequence of pixel flips that each

1. preserve a set of desired topological properties

2. occur only at locations within a mask M,

then we will say L is a *warping of* L^*, or $L \lhd L^*$. The first condition constrains L and L^* to be topologically equivalent. The second condition can be used to constrain L to be geometrically similar to L^*. We will write simple(L) to denote the set of simple points of L.

Definition 7.1. The *warping error* between some candidate labeling T and a reference labeling L is the size of the difference set $E(L,T) = L\Delta T = W_L \backslash W_T \cup W_T \backslash W_L$ for the "best warping" of L^* onto T:

$$D(T||L^*) = \min_{L \lhd L^*} |E(L,T)| \tag{7.1}$$

In Figure 7-2 we illustrate the warping error and compare it to standard pixel error.

7.3.1 Homotopic digital warping algorithm

Finding the global minimum of the cost function in Eq. 7.1 is an NP-hard problem and therefore not a practical goal for realistic datasets. Instead, we introduce an algorithm for computing local minima of the warping error.

We start by initializing L at L^*. Then we flip a random point difference $p \in E(L,T)$ in L, to reduce $|E(L,T)|$. The location of the flipped label is randomly chosen, but must be a simple point of L to ensure that the warping L remains topologically equivalent to L^*. This label relaxation is repeated until there are no further simple locations in $E(L,T)$. This process will always converge, but it may find a local minimum of $D(T||L^*)$. The procedure is described in pseudocode in Algorithm 7.1.

The existence of local minima within the computation of a metric is a legitimate concern. However, we note that this is also an issue with the Berkeley segmentation benchmark, which does not produce deterministic results due to the use of random nodes and edges in the sparse graph constructed to perform the boundary matching (see Section 7.2.2 for further details).

Algorithm 7.1 Descent algorithm for warping a binary image L^* to an analog image T, under geometric constraints set by the binary image M. We write simple(L) to denote the set of simple points of L.

warp($L^* \in \mathcal{B}$,$T \in \mathcal{A}$,$M \in \mathcal{B}$)
$L := L^*$
do
    $S :=$ simple$(L) \cap M$
    $i := argmax_{j \in S} |t_j - l_j|$, breaking ties randomly
    if $|t_i - l_i| > 0.5$
        $l_i := 1 - l_i$
    else
        return L
    end

The development or more sophisticated warping algorithms that result in more consistent or lower-error results is an interesting avenue for research; previous work in homotopic thinning algorithms may suggest certain strategies (for example, using the distance transform to determine flipping order [83]). Empirically, however, even an algorithm based on random flipping yields highly reproducible results.

7.3.2 Geometric constraints

In practice, we may want to limit the amount of geometric distortion that L is allowed to achieve. To implement a geometrical constraint, a mask M is created from the intersection of the dilation of L^* and the dilation of the complement of L^*. Dilation is a standard operation in mathematical morphology that expands the borders of "1" regions. Therefore the mask contains only pixels that are near borders between "1" and "0" regions. During the relaxation, only simple points within the mask are altered. As a result, when using a geometric constraint the difference set may include simple points of L that lie outside the mask. At each iteration of label relaxation, we consider locations that are simple points, are contained in the mask M, and at which L and T disagree.

7.3.3 Warping error: boundary detection or segmentation metric?

At first glance, it may seem that the warping error measures only boundary detection performance. But we would like to argue that it is also a good measure of segmentation per-

formance. This is because digital topology tells us how any single pixel affects the global topology of an image. Let L be an optimal warping (in terms of minimizing Eq. 7.1) of L^* onto T. Any pixel in the symmetric difference set $L\Delta T = L\backslash T \cup T\backslash L$ represents a *topological error* in T because flipping its value in L (which is topologically equivalent to the ground truth L^*) would cause a topological change. Thus the warping error is an upper bound on the number of topologically-relevant boundary labeling errors in T (if a geometric mask is used, then the warping error also includes labeling errors of a geometric nature). Therefore, if segmentations are generated from T and L^* by finding their connected components, then the warping error should be a reasonable measure of the topological disagreements between the segmentations. Moreover, using the classification scheme for non-simple points in Table 6.1, we can classify the particular type of topological change that any single labelling error is associated with (split, merger, object addition, etc).

The Rand index is becoming more popular as a metric of segmentation performance. It can be used to compare segmentations in which regions are non contiguous clusters of pixels. Such segmentations are not equivalent to boundary labellings, so the warping error cannot be applied. In many applications, such as the one studied below, this is not a significant limitation.

The warping error can be distinguished from the Rand index in other respects. The warping error can penalize all kinds of topological errors, including the presence of holes and handles, but the Rand index penalizes only connectivity errors. In certain medical imaging situations, control of such aspects of topology is especially important [36]. The Rand index mildly penalizes shifts in boundary location, while the warping error ignores them. The warping error weights a topological error by the number of pixels involved in the error itself, while the Rand index weights a split or merger by the number of pixels in the objects associated with the errors.

The warping error provides a way to relate a segmentation to its underlying representation as a boundary or in/out map. However, maintaining this relationship imposes certain restrictions. For example, only pixels that are κ-connected in the digital image can be connected in the resulting segmentation (the Rand index, in contrast, makes no assumptions about the *origin* of connectivity and accepts an arbitrary clustering of points). In practice,

however, it is not clear that this is a significant limitation. Typically we are interested in interpreting images in which a region is defined by some physically contiguous region that corresponds to a collection of adjacent pixels in the image.

Chapter 8

Optimizing Warping Error with BLOTC

Abstract. In this chapter we show how the warping error metric introduced in Chapter 7 can be used as a cost function in the supervised learning of boundary detection. We discuss conceptual reasons why such an approach should lead to superior results, and empirically demonstrate the advantages of the technique on the challenging problem of segmenting electron microscopy images of brain tissue. This work is one of the first applications of digital topology to the area of machine learning.

8.1 Introduction

The accurate detection of object boundaries in images has been a long-standing challenge for computer vision. Recently, a supervised learning approach has become popular, promoted by the creation of the Berkeley Segmentation Dataset, a collection of natural images along with ground truth boundaries traced by humans [69]. Accuracy of boundary detection has been improved relative to classic algorithms by training simple classifiers that combine hand-designed features [69, 70, 71]. Combinations of large numbers of simple features have been learned using boosting [25], and convolutional networks have been trained to perform boundary detection directly on raw images taken from biological microscopy, with no use of hand-designed features (an approach described in Chapter 5 of this thesis). Multi-scale image cues have also been incorporated to provide greater context in boundary estimates [87].

Supervised learning of boundary detection requires a quantitative measure of performance with which to compare machine boundary labellings with human boundary labellings. This measure is then used as a cost function that is subject to optimization during learning. Traditionally, supervised approaches to boundary detection have relied on pixel error as the cost function. However, some approaches have employed simple fixes to address the arbitrary nature of ground truth boundary locations. For example, Martin et al. [71] blur the human boundaries, which has the disadvantage of permitting multiple detections of the same boundary. In [25] the authors averaged multiple human labellings to yield an estimate of the probability that a boundary exists at each location.

We find these fixes unsatisfactory, and argue that supervised learning of boundary detection demands a more sophisticated error measure that truly measures the topological accuracy of the boundary detector. We believe that choosing and optimizing the correct cost function is not a minor technicality, but a major barrier to attaining good performance of boundary detection in the supervised learning approach.

The warping error we introduced in Chapter 7 is a good candidate for the correct cost function. Firstly, it tolerates errors in boundary location, but retains sensitivity to topological errors. Second, it can be used directly as a cost function for learning, which leads to a method we call Boundary Learning by Optimization with Topological Constraints (BLOTC). When warping error is optimized, small errors in boundary localization are eliminated from the cost function for learning. This leaves only true segmentation errors: places where objects have been split, merged, or other topological changes have occur ed (such as object addition or deletion). Consequently BLOTC training focuses the learning process on locations in the image that are critical for accurate segmentation.

As a demonstration, we apply BLOTC to learning the detection of boundaries in electron microscopic images of neural tissue. Boundary detectors are trained using both BLOTC and standard methods. When evaluated on a test set using the warping-based error measure, the BLOTC detector significantly outperforms the standard detector.

Finally, we also apply BLOTC to the task of interactive segmentation, finding regions based on a sparse subset of seed points provided by a human operator.

8.2 Boundary learning by optimization with topological constraints (BLOTC)

Our goal is to find a function that maps an image patch to an estimate of the probability that the central pixel is labeled "1." We will call such a function an image patch *classifier*. If the classifier is applied to patches centered at all locations in an image, it produces an output image that is called a *boundary map* or an *in-out map*, depending on convention (see Fig. 6-6). The analog values in this probability map can be thresholded to produce a binary image labeling. We will assume that both the classifier output and ground truth are represented as in-out maps.

Consider a binary machine labeling T and a human labeling L^* of the same image. When optimizing pixel error, we would ideally like to minimize the number of binary disagreements between T and L^*: $\sum_i \delta(t_i, l_i^*)$ where the machine and human labeling at location i are denoted by t_i and l_i^*, respectively.

However, it is often easier to optimize a smooth cost function that depends on the real-valued output of the classifier. Let $F_I(\vec{w})$ be the analog probability map produced by applying the classifier to the entire image I. At each location, the probability map is an analog value between 0 and 1, representing the classifier's estimate of the probability that the label of the image pixel is "1." Since the probability map $F_I(\vec{w})$ and a binary labeling L of the image I are both images of the same size, we can compute their Euclidean distance:

$$d(F_I(\vec{w}), L) = \sum_i [l_i - f_i(\vec{w})]^2$$

where the probability map at location i is written as $f_i(\vec{w})$. Here the dependence of $f_i(\vec{w})$ on the image I is left implicit for notational convenience. Optimizing the Euclidean distance thus serves as an approximation to optimizing the binary classification error when $F_I(\vec{w})$ is thresholded to yield a binary map.

In BLOTC our goal is to minimize the warping error, defined in Equation 7.1. We generalize the warping error to compare the analog classifier output $F_I(\vec{w})$ with the human labeling L^*,

Algorithm 8.1 Stochastic online gradient learning. The learning rate parameter η is small and positive.

gradient($I \in \mathcal{A}$,$L \in \mathcal{B}$,$\vec{w}$,k)
for iter = 1 to k
    i =random location in random image
    $\vec{w} := \vec{w} - \eta \nabla_{\vec{w}} |l_i - f_i(\vec{w})|^2$
end
return $\vec{w}$

$$D(F_I(\vec{w}) > \theta || L^*) = \min_{L \lhd L^*} |E(L, F_I(\vec{w}) > \theta)| \tag{8.1}$$

in which the analog network output is thresholded at some fixed value of θ to yield a comparison between binary-valued image labelings.

For supervised learning, we would like to minimize Eq. 8.1 with respect to the classifier parameters $\vec{w}$. In order to make this easier to optimize, we again use a smooth approximation of the binary error:

$$\min_{\vec{w}} D(F_I(\vec{w}) || L^*) = \min_{\vec{w}} \min_{L \lhd L^*} d(F_I(\vec{w}), L)$$

in which we have defined the warping error between an analog valued quantity and a binary image labeling using the Euclidean distance. We call this method Boundary Learning by Optimization with Topological Constraints, or BLOTC. Note that standard training is the case where no warping of L^* is allowed, i.e., the geometric constraint becomes completely tight.

The warping of L^* onto $F_I(\vec{w})$ is carried out in much the same way as described in Chapter 7. The only difference is that the label locations satisfying the topological and geometrical constraints can be rank ordered using $|l_i^* - f_i(\vec{w})|$, so that the locations of flipped pixels can be chosen greedily rather than randomly (Algorithm 7.1). The minimization with respect to $\vec{w}$ is carried out using the usual method (e.g., gradient descent). The dual optimization alternates between updates of L and updates of $\vec{w}$ (Algorithm 8.2).

Algorithm 8.2 Boundary Learning by Optimization with Topological Constraints (BLOTC)

blotc($I \in \mathcal{A}$,$L^* \in \mathcal{B}$,$M \in \mathcal{B}$,k_1,k_2)
$L := L^*$
$\vec{w}$:=random initialization
$\vec{w}$:=**gradient**(L,$\vec{w}$,k_1)
repeat
    L :=**warp**(L,$F_I(\vec{w})$,M)
    $\vec{w}$:=**gradient**(I,L,$\vec{w}$,k_2)
until convergence
return $\vec{w}$

8.3 Results

We performed two separate experiments to test the BLOTC approach to learning boundary detection. In the first experiment the goal was to perform a 2d segmentation of conventionally stained electron microscopy images of mouse hippocampus obtained using ATLUM [37]. Although the dataset is 3d, the extremely anisotropic resolution of the data makes a fully 3d approach less natural. We tested a fully 3d approach on images of rabbit retina obtained using SBF-SEM, which yields lower resolution images in the x-y plane but much more isotropic datasets overall [24].

Beyond resolution, another major difference between the two datasets is the staining that was used to prepare them prior to imaging. The mouse hippocampus images were conventionally stained, meaning that both intracellular organelle (such as mitochondria and synaptic vesicles) and cell membranes are visible in the resulting images. This poses a potential challenge for computational techniques, which must not let the presence internal organelle interfere with identifying true neurite boundaries. This concern motivated the specialized extracellular staining technique used in the SBF-SEM images [18], which renders only cell membranes visible.

8.3.1 2d segmentation of ATLUM images

Serial sections of mouse hippocampus were obtained using the Automated Tape Collecting Lathe Ultra-Microtome (ATLUM) [37]. Sections were 29.8 nm thick, and images of each section were taken at roughly 5nm/pixel using a JEOL scanning electron microscope. The

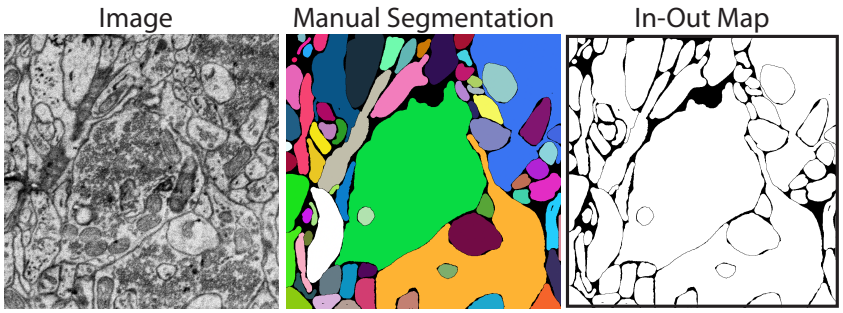


Figure 8-1: A 1024×1024 pixel image of mouse hippocampus, obtained using ATLUM, along with a human segmentation and the corresponding in-out map. Clutter from intracellular structures within cells makes human interpretation challenging, but the (high) resolution of the image makes most boundaries ultimately unambiguous even from viewing a single plane.

images were downsampled to roughly 20nm/pixel for analysis (see Figures 8-1 and 8-2 for examples of the EM images). Our goal in this section was to segment the images into regions corresponding to the cross sections of distinct neurons. This is a first step toward full 3d reconstruction of neuron shapes.

The segmentation task is especially challenging because the neurons contain many intracellular organelles such as mitochondria, and this internal clutter can be distracting. Two kinds of boundaries are visible in the images: external boundaries between neurites, and internal boundaries of the intracellular organelles. There is an obvious question concerning how the boundary detector should be trained. We definitely want the external boundaries of neurons to be detected. But do we want the internal boundaries to be detected?

It turns out that this decision can be left up to the computer, if BLOTC training is used. We implemented a version of BLOTC in which some topological changes were allowed in the warping described in Section 2.1. In addition to flipping simple pixels, the warping was allowed to flip certain types of non-simple pixels (non-simple pixels can be rigorously classified; see supplementary). In particular, the warping was allowed to create holes within objects, not penalizing the computer for detecting internal boundaries, even if they were not traced by the human.

A 256×256 bounding box from a set of 100 aligned images was cropped from the dataset to form a $256 \times 256 \times 100$ image stack. Each image in the stack was manually traced by a human using the ITK-SNAP software package. All external boundaries of neurons were

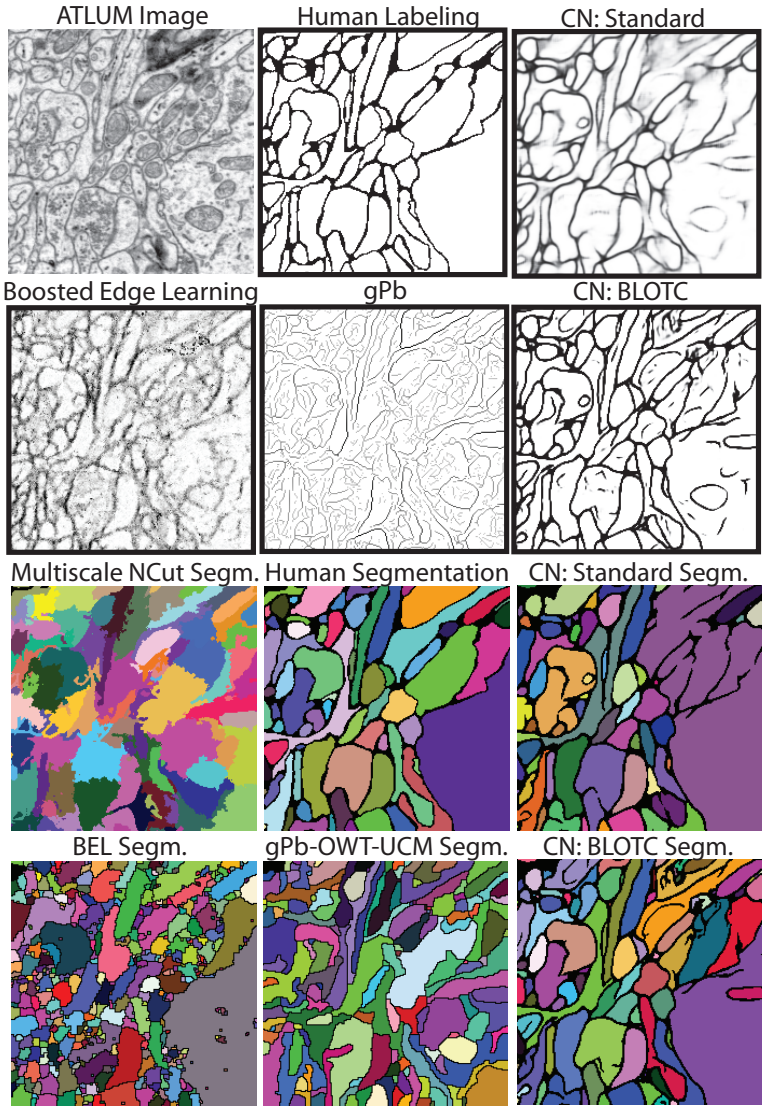


Figure 8-2: Visual comparison of output from a convolutional network (CN) trained in the standard way and a CN trained with BLOTC, on an image from the test set. Both Standard and BLOTC CN segmentations were generated by connected components on the respective boundary detector outputs at threshold 0.75. For gPb and BEL, segmentations were generated at the optimal threshold according to the Rand index. Multiscale NCut directly generates a segmentation and thus no threshold was used (in the results shown above, the true number of objects in this specific test image was provided as input to to the multiscale normalized cut routine). The BLOTC CN segmentation has a substantially smaller number of split and merger errors compared to the CN trained in the standard way.

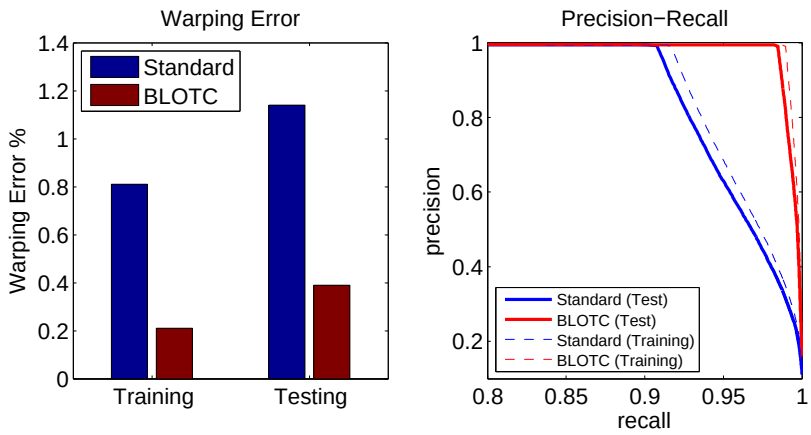


Figure 8-3: Comparison of standard and BLOTC learning using the warping error metric on a 5.2 megavoxel training set and a 1.2 megavoxel testing set. The left plot shows warping error when classifier output is thresholded at 0.5, demonstrating a large relative reduction in error from standard to BLOTC training. The right plot shows the precision-recall on outside voxel classification accuracy, which compensates for the class-imbalance, as outside pixels are the more rare class.

traced in each image, but the internal boundaries were mostly neglected. Figure 8-2 shows an example of human tracing. From this dataset, 80 images were used as a 5.2 megavoxel training set and 20 images were set aside as a 1.3 megavoxel testing set.

For our image patch classifier, we used a convolutional network. It has already been shown that convolutional networks provide state-of-the-art performance at boundary detection in EM images [44], when trained by standard methods. Here we show that BLOTC training improves performance even more. However, it is important to note that BLOTC is not limited to convolutional networks, but can be used to train any kind of classifier.

We trained two convolutional networks with identical architectures containing 6 hidden layers, 24 feature maps in each hidden layer, and full connectivity between feature maps in adjacent layers. Each individual filter was 5×5 pixels, but the multiple-layers yield an effective field of view of the classifier of 28×28 pixels. The standard network was trained for 1,000,000 updates using the traditional optimization with the labels fixed. The BLOTC network was initialized with the weights from a standard network after 500,000 gradient updates and then further trained for 500,000 additional updates using BLOTC optimization. Each network was trained in roughly 18 hours, using a layer-wise procedure that iteratively adds hidden layers to the network architecture. The training used a fast

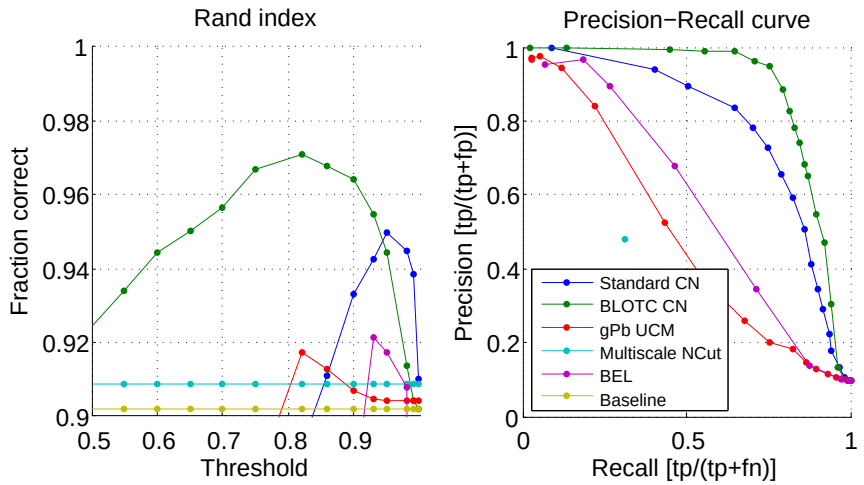


Figure 8-4: Comparison of standard and BLOTC learning using the Rand index and precision-recall of correctly classified connected pairs of pixels on a 1.2 megavoxel testing set. The relative reduction in Rand error between standard and BLOTC results is large (40%). The precision-recall curve compensates for the class imbalance in pixel connectivity (most pixels are disconnected from one another). gPb UCM corresponds to segmentations generated by regions created by an oriented water transform on gPb contour detector output, followed by conversion into a hierarchical region tree [3]. Multiscale Ncut corresponds to multiscale normalized cut [23]. This method requires as input the number of objects in an image; we provided the average number of objects in a training set image. Boosted Edge Learning (BEL) was learned using the same 5.2 megavoxel training set as the convolutional networks, with a 30×30 field of view and identical classifier parameters to those used in [25] with code provided by the authors. Several methods for generating segmentations from BEL output were tested; a watershed approach worked best. Baseline corresponds to a segmentation in which all pixels are disconnected from one another.

shared-memory GPU implementation that provides between a 50-100× increase in training speed as compared to CPU implementations.

The results of training are shown in Figure 8-2. Both BLOTC and standard networks do a good job of detecting boundaries between neurons, and ignore most intracellular structures such as vesicles. The BLOTC network strongly detects some mitochondrial boundaries that were not in the human tracing, presumably because it was not penalized for doing so. The standard network cannot manage to ignore mitochondrial boundaries, even though it was trained to ignore them. This is presumably because mitochondrial boundaries often resemble external boundaries, at least locally. The BLOTC network produces more binary output, as if it were more confident.

Figure 8-3 shows that the warping error of the BLOTC network is much lower than that

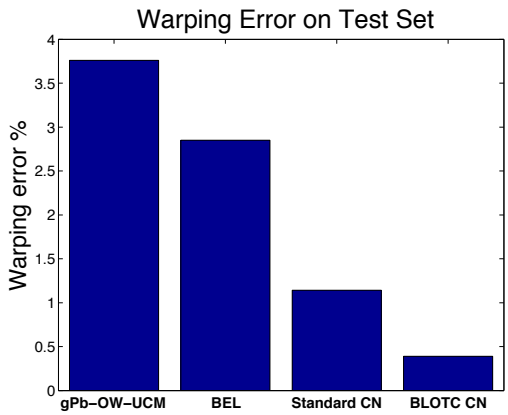


Figure 8-5: Warping error on a 1.2 megavoxel test set. BEL and convolutional network methods were trained on a corresponding 5.2 megavoxel training set. For this comparison, a threshold of gPb-OW-UCM and BEL was chosen according to the threshold that achieved highest Rand index also on the test set (shown in Figure 8-4). These results are consistent with the relative ordering of algorithms that the Rand index produced, but the relative reduction in error between the methods is larger (for example, the gPb-OW-UCM method has almost ten times as much warping error as the highest performer, BLOTC CN).

of the standard network. The value of the error is low (1% or less), which is primarily a consequence of the fact that boundary pixels are relatively rare. In general, the total error has a limited dynamic range in a classification problem with a highly skewed class distribution. Instead it is helpful to view performance using precision-recall curves, which also demonstrate the superior performance of BLOTC, as shown in 8-3.

Figure 8-4 demonstrates the superiority of the BLOTC network using the Rand index to quantify segmentation performance. The BLOTC network again outperforms the standard network. The value of the Rand index is high (greater than 90%). As before, this is because the class distribution is skewed. In the ground truth, most pairs of pixels belong to different objects. Therefore the trivial segmentation in which every pixel belongs to a different object has a Rand index of over 90%. This sets the baseline for performance in Figure 8-4. We also benchmarked several leading competitors: multiscale normalized cuts [23], gPb-OW-UCM [3], and Boosted Edge Learning (see supplementary for additional information regarding use of these methods). Their performance was much worse than that of our convolutional networks, and only barely above the baseline of the trivial segmentation.

Figure 8-4 also exhibits precision-recall curves for connected pixel pairs, which again demonstrate that the BLOTC network is superior to the standard network. Perhaps it is

not surprising that gPb-OW-UCM and multiscale normalized cuts are so inferior, as they were designed for natural images, while our networks have been optimized for EM images. But Boosted Edge Learning was also trained on the EM images, but its performance is still closer to that of gPb-OW-UCM than our networks.

In summary, these results provide strong evidence that BLOTC can significantly improve the performance of boundary detection, even on approaches that already obtain high performance (such as the standard convolutional network we train).

8.3.1.1 Details of Experimental Procedures

In this section we provide additional details of the experimental comparisons that were performed on 2d ATLUM images.

Multiscale Normalized Cut

Multiscale normalized cut was performed using publicly available code provided by the authors of [23]: http://www.seas.upenn.edu/~timothee/software/ncut_multiscale/ncut_multiscale.html

This technique requires that the number of objects in the image be specified by the user. We are interested in completely automated segmentation in which such information would not usually be available. Therefore we provided to the code the average number of objects in a training set image. However, we also tried providing the code with the true number of objects in each test set image. Then the results of "Multiscale Ncut" shown in Figure 4 of the main text improve slightly, but remain worse than all other techniques.

gPb-OWT-UCM

The gPb-OWT-UCM algorithm (global probability of boundary followed by the oriented watershed transform and a hierarchical region construction by ultrametric contour maps) was performed using publicly available code provided by the authors of [3, 66]: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/gpb/grouping.zip

The following is a summary of the algorithm implemented by the code, as described in [3]. First, the gPb contour detector was applied directly to the raw EM images, producing an 8-channel oriented localized probability of boundary map, as well as a single-channel

thinned contour image which is the result shown in Figure 2 of the main text. The 8-channel boundary map was then converted to an oversegmentation using the oriented watershed transform, and then an ultrametric contour map according to the dissimilarity between regions as determined by mean probability of boundary value. Finally, the ultrametric contour map was partitioned using connected components at various thresholds (where the x-axis in Figure 4 for this technique corresponds to a segmentation of the UCM thresholded at a value of $255 * (1 - x)$, to compensate for the range of the UCM, and then the binary boundary map is converted to an in/out map prior to connected components by flipping the binary values).

Boosted Edge Learning

The Boosted Edge Learning algorithm was applied to our training set of EM images using publicly available code provided by the authors of [25]: http://www.loni.ucla.edu/~ztu/Download.htm

A probabilistic boosting tree of depth 10 was used with 120 weak classifiers in each AdaBoost node. A patch size of 30 was used, which is a slightly larger than the field of view of the convolutional networks that were used. These were the default parameters provided in the code.

After training was complete, we proceeded to quantify segmentation performance on the test set of images. Here we had to choose our own method for generating segmentations from BEL output. The simple procedure of finding connected components of the thresholded BEL output (which was used for the CNs and gPb-OWT-UCM) produced a poor Rand index, as did the watershed transform on the negative of the BEL output. Therefore we applied the watershed transform only after using MATLAB's `imimposemin` command to damp local maxima of the BEL output, constraining them to occur where the BEL output was greater than a threshold. The watershed transform was performed with 8 connectivity (4 connectivity gave worse results).

Convolutional Networks

As our loss function during optimization of the convolutional network, we used the square-square loss

$$l(x, \hat{x}) = x \max(0, 1 - \hat{x} - m)^2 + (1 - x) \max(0, \hat{x} - m)^2$$

with $m = 0.2$, rather than the squared loss $(x - \hat{x})^2$. This loss function is a better approximation of the true binary classification error we seek to optimize. This is particularly important in the scenario in which the classifier has predicted the correct class of most examples (based on thresholding the analog output), and there are relatively few remaining incorrectly classified examples. In this case, the small amount of remaining error in the squared loss related to pushing values to 0 or 1 may dominate the error associated with incorrectly classified examples. The square-square loss allows the training to "give up" when the classifier output is correct by a sufficient margin. This gives the classifier more flexibility to achieve higher binary classification accuracy.

Batch learning is inefficient in this context, as the training set has millions of pixels and it is not practical to compute the gradient with respect to the entire training set for each update, particularly when many hundreds of thousands of updates may be required in order to reach convergence. Therefore we adapted stochastic online learning to this problem. We employed a minibatch implementation in which a randomly chosen 14×14 patch of the network output was used to compute each gradient update. A localized patch shares computation in a convolutional network and is therefore especially efficient to compute.

Segmentations were generated by thresholding the analog output of the convolutional network and then performing connected components with the same $\kappa = 4$ and $\bar{\kappa} = 8$ adjacency using in warping.

8.3.2 3d segmentation of SBFSEM images

We also tested BLOTC on a dataset of electron microscopy images of rabbit retina, collected using Serial Block Face Scanning Electron Microscopy [24]. This dataset is similar, although not identical, to the one studied in Chapter 6. The images were collected at a resolution of 22nm/pixel x-y resolution with a sectioning thickness of roughly 30nm. This nearly isotropic

resolution enabled us to pursue truly 3d boundary detection. Therefore both the standard and BLOTC methods boundary detectors were trained using 3d patch inputs, and 3d digital topology was used during warping computations.

Two separate $100 \times 100 \times 100$ voxel datasets were cropped from the image volume. Humans used a custom software tool to trace objects manually, which provided both object boundaries and consistent object identities throughout a 3d volume. These tracings were used to label the voxels as either intracellular or extracellular, generating a binary in-out labeling. One volume was used as a training set, and the other was used as a testing set. Each volume contained roughly 90 distinct objects. When two human in-out labellings of the same image volume are compared with each other, they disagree on about 10% of the voxels. The vast majority of disagreements are about the precise locations of object boundaries, not about the identities of objects.

We used a convolutional network as our image patch classifier. We trained two convolutional networks with identical architectures containing 2 hidden layers and 5 feature maps in each hidden layer. Each of the 35 filters was $5 \times 5 \times 5$, for a total of over $4,000$ parameters adjusted by learning. The 'standard' network was trained for $500,000$ updates tushing the traditional optimization with the labels fixed. The BLOTC network was initialized with the weights from a standard network after $350,000$ gradient updates and then further trained for $100,000$ updates using BLOTC optimization (see Supplementary Material for full details). The results are shown in Figure 8-6. Even to casual inspection by eye, the output of the BLOTC network looks superior to that of the standard network. The boundaries between objects are clearer, especially at the locations of several potential mergers between objects. At certain locations, even the BLOTC labeling looks superior.

For a more objective comparison of the two methods, we thresholded the in-out maps and found their connected components. We counted the split and merge errors in the resulting segmentations, using the metric described in section 5. The most important graph in Figure 8-6 is the number of mergers vs. the number of splits for all possible choices of threshold. This graph shows that the BLOTC network is markedly superior to the standard network. For any segmentation based on the standard network at some threshold, there exists a segmentation based on the BLOTC network at some threshold for which both the number

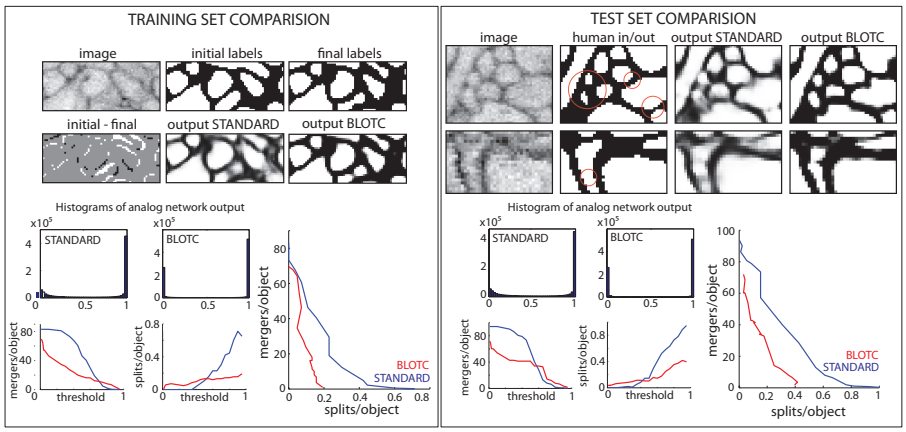


Figure 8-6: Results on 3d segmentation of SBFSEM images.

of splits and the number of mergers is reduced.

The improvement in segmentation performance occurs despite a *decrease* in performance as measured by voxel error according to the initial human labeling. On the training set, the standard network attained a final voxel error of 7.5%, while the BLOTC network had a final voxel error of 9.2%. The higher voxel error of the BLOTC network is due to the fact that about 12% of the original human labels were flipped to arrive at the target labels. On the testing set, the standard network had a final voxel error of 10.34%, and the BLOTC network had a final voxel error of 10.7%.

When voxel error is computed using the warping metric defined in section 2, we observe a performance increase consistent with that observed in the split-merger curve. After warping, the remaining voxel differences correspond to topological errors in the network output. On the training set, the BLOTC network attains a warped voxel error of 0.12%, while the standard network warped voxel error is 0.17%. For the testing set, the BLOTC warped voxel error is 0.19% as compared to 0.26% for standard training. The fractional decrease of warped voxel error is large, and is presumably due to the reduction of splits and mergers. Combined with the split-merger results, these measurements provide strong evidence that BLOTC can significantly improve the topological accuracy of a learned boundary detector.

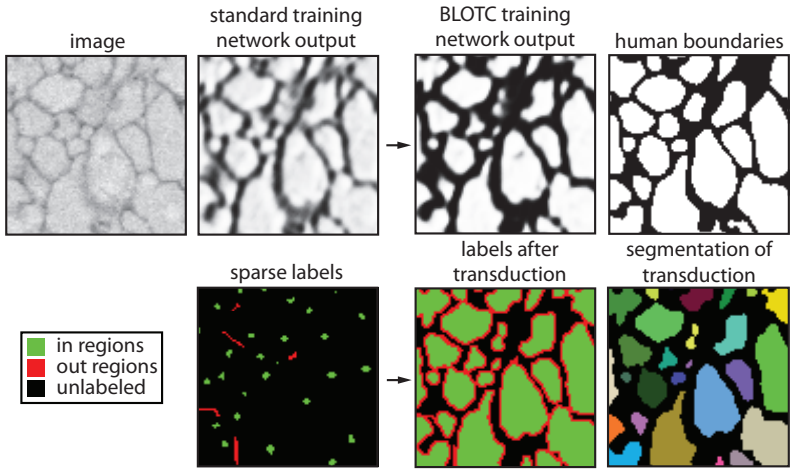


Figure 8-7: Interactive segmentation from seed points.

8.3.3 Interactive segmentation and semi-supervised learning

In this section we show that BLOTC can be used for interactive segmentation, in which the user labels a small subset of seed points in an image and the task of the computer is to determine the full regions [92]. Our seed points were an extremely sparse labeling of a single 2d SBFSEM slice, shown in Figure 4. Using only this sparse labeling (full human tracing shown only for reference), a two-layer convolutional network was trained in the standard way, and then applied to the full image. Unsurprisingly, the results are are poor due to the small amount of labeled data. Another network with identical architecture was also trained from the sparse labeling, but using BLOTC (see Supplementary Material for details).

The results shown in Figure 4 are interesting in two ways. First, the label relaxation in BLOTC causes the sparse labels to dramatically grow in size and produce a plausible interpretation of the full image. This illustrates how BLOTC can be used for interactive segmentation, which is also known as transductive segmentation [26]. Second, the output of the BLOTC optimized network is considerably better than the standard network output, even though both networks use the *same exact* sparse labeling. Because BLOTC iteratively expands the training set using the simple point criteria, the final result yields a superior classifier. This sort of training can thus be considered a form of semi-supervised learning, in which labels are propagated to unlabeled examples.

Chapter 9

Conclusions and Outlook

In this thesis we have advanced a new computational approach to low-level vision. The first aspect of this approach is the use of convolutional networks that operate directly on an image and are completely adapted towards a specific image processing task. This is a significant departure from either completely hand-designed methods or learning methods that rely on a hand-designed feature space. We have demonstrated that this approach can yield success in two very different domains: electron microscopy images of brain tissue and natural images.

We expect that the power of this approach will increase with the ability to explore larger models due to advances in computational power. For example, an obvious extension of our approach is to introduce additional structure within the convolutional network architecture to analyze the image at multiple scales (this requires both subsampling to downsample the image and supersampling to project the coarse-scale analysis back to the desired resolution). This would enable the classifier to take advantage of much greater image context in the prediction of any single output value. Advances in learning strategies such as unsupervised and semi-supervised initialization of multi-layer networks [42, 86, 4] may also enhance the quality of the learned parameters. Finally, the success of the basic convolutional network strategy in low-level vision analysis of natural images and electron microscopy images suggests possible application to a variety of other domains and problems (e.g., medical imaging).

We have also shown that as an architecture for computation, convolutional networks are mathematically related to another popular approach (Markov Random Fields), but avoid certain difficulties in learning and inference that occur in methods which rely on probabilistic

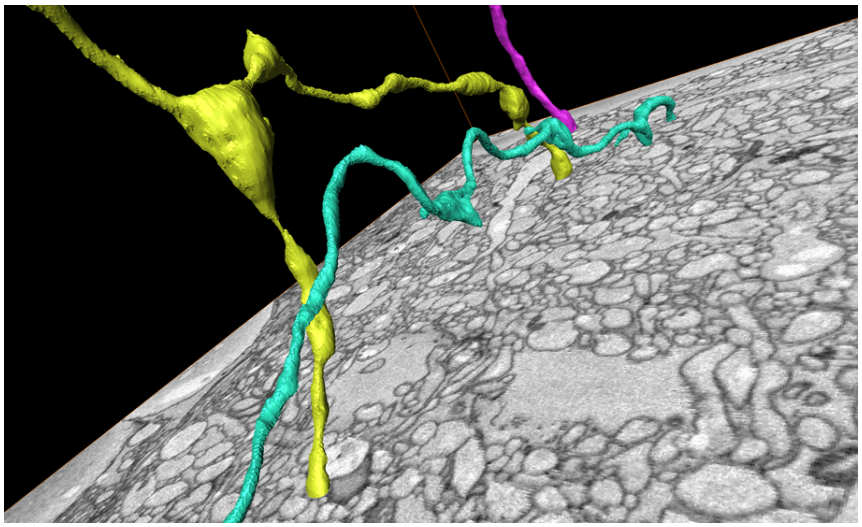


Figure 9-1: Putative neurites automatically traced using techniques described in this thesis, in a volume of rabbit retina imaged with Serial Block Face Scanning Electron Microscopy (SBF-SEM) [24, 18].

modeling. The specific relationship established in this respect suggests that Markov random fields are closely related to *recurrent* convolutional networks, in which each layer has the same weights (except the first and last). An interesting question is whether the networks we learn without constraining each layer to have the same weights are approximating a recurrent network, or are truly learning some sort of hierarchical composition of feature representations. In general, it is desireable to achieve a greater theoretical understanding of multi-layer networks; recent work has taken initial steps along these lines [98].

The second aspect of our approach has been the use of domain specific cost functions and learning algorithms that reflect the structured nature of certain prediction problems. In particular, segmentation and boundary detection of a single input (i.e., a single image) involves the prediction of many correlated variables: the presence or absence of a boundary at each location in the image. Although it is possible to adapt traditional strategies to such problems by treating all such predictions as essentially independent, we have shown that this results in suboptimal performance. For the case of boundary detection, we have rigorously grounded the problem within the the framework of digital topology and shown how major concepts from digital topology can be used to solve critical problems in this domain, such as defining an appropriate error metric and optimization strategy. This has

resulted in BLOTC, the first learning algorithm that optimizes the *topological* accuracy of a set of correlated output variables. As part of this goal, we have also provided a novel and complete characterization of non-simple points based on the concepts of topological number and extended topological number.

We believe these are important steps forward for the machine learning approach to boundary detection and segmentation, but there remains significant room for progress. In particular, BLOTC is but a first step towards optimizing the high-order property of topology. The basic concept of a constrained warping of ground truth is simple and reasonably effective, but it would be interesting to realize ways of more directly integrating topological constraints into the gradient learning procedure.

Finally, the methods introduced in this thesis represent a substantial advance in the image analysis methods required for connectomics. The availability of methods that can operate with little prior knowledge is an advantage in this context as there are a variety of different imaging modalities that are being pursued, each with unique low-level characteristics. A machine learning strategy that can learn directly on the image space without having to specify features or prior knowledge in advance is therefore a significant advantage. More important than convenience, however, is that this approach can lead to superior accuracy. At this point, none of the described methods (nor any other in the literature) can achieve the levels of accuracy required to "solve" the image analysis problems in connectomics. However, we are optimistic that with continued efforts in improving classifier architectures as well as learning strategies, automated approaches may soon enable semi-automated reconstructions of substantial and interesting neural circuits.

Bibliography

[1] D. Ackley, G. Hinton, and T. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 1985. 69, 79

[2] Björn Andres, Ullrich Köthe, Moritz Helmstaedter, Winfried Denk, and Fred A. Hamprecht. Segmentation of sbfsem volume data of neural tissue by hierarchical classification. In *Pattern Recognition*, volume 5096, pages 142–152, 2008. 21

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *Proc. CVPR*, 2009. 13, 105, 119, 120, 121

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy Layer-Wise Training of Deep Networks. *NIPS* 2006*. 33, 52, 127

[5] Y. Bengio and Y. Le Cun. Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, 2007. 34

[6] M. Bertero, T.A. Poggio, and V. Torre. Ill-posed problems in early vision. *Proceedings of the IEEE*, 76(8):869–889, 1988. 16

[7] G. Bertrand. A Boolean characterization of three-dimensional simple points. *Pattern recognition letters*, 17(2):115–124, 1996. 88

[8] G. Bertrand and G. Malandain. A new characterization of three-dimensional simple points. *Pattern Recognition Letters*, 15(2):169–175, 1994. 88, 89, 95

[9] J. Besag. Spatial Interaction and the Statistical Analysis of Lattice Systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(2):192–236, 1974. 37

[10] J. Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, 1977. 68, 79

[11] M.J. Black, G. Sapiro, D.H. Marimont, and D. Heeger. Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, 7(3):421–432, 1998. 45

[12] Benjamin Bollmann. Image segmentation using supervised learning with topologically constrained label relaxation. Master's thesis, ETH Zürich, 2008. 21, 22, 99

[13] L. Bottou and Y. LeCun. Large scale online learning. *NIPS* 2003*. 30, 77

[14] L. Boxer. A classical construction for the digital fundamental group. *Journal of Mathematical Imaging and Vision*, 10(1):51–62, 1999. 85

[15] L. Boxer. Properties of digital homotopy. *Journal of Mathematical Imaging and Vision*, 22(1):19–26, 2005. 84, 85

[16] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 2004. 69, 73, 78

[17] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23, 2001. 69, 73, 78

[18] Kevin L Briggman and Winfried Denk. Towards neural circuit reconstruction with volume electron microscopy techniques. *Curr Opin Neurobiol*, 2006. 7, 14, 19, 20, 61, 63, 115, 128

[19] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986. 16, 17

[20] D. Chandler. Introduction to modern statistical mechanics. *New York: Oxford*, 1987. 40, 42

[21] B.L. Chen, D.H. Hall, and D.B. Chklovskii. Wiring optimization can relate neuronal structure and function. *Proceedings of the National Academy of Sciences of the United States of America*, 103(12):4723, 2006. 20

[22] BV Cherkassky and AV Goldberg. On Implementing the Push—Relabel Method for the Maximum Flow Problem. *Algorithmica*, 19(4):390–410, 1997. 104

[23] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, page 1124. Citeseer, 2005. 13, 119, 120, 121

[24] Winfried Denk and Heinz Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biology*, 2004. 7, 14, 19, 20, 63, 115, 123, 128

[25] P. Dollar, Z. Tu, and S. Belongie. Supervised Learning of Edges and Object Boundaries. In *CVPR*, 2006. 13, 23, 111, 112, 119, 122

[26] O. Duchenne, J.Y. Audibert, R. Keriven, J. Ponce, and F. Segonne. Segmentation by transduction. In *CVPR*, 2008. 126

[27] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networks- A review. *Pattern Recognition*, 35, 2002. 23, 75

[28] X. Feng, C.K.I. Williams, and S.N. Felderhof. Combining belief networks and neural networks for scene segmentation. *PAMI*, 24(4):467–483, 2002. 69

[29] S. Fourey, T.Y. Kong, and G.T. Herman. Generic axiomatized digital surface-structures. *Discrete Applied Mathematics*, 139(1-3):65–93, 2004. 99

[30] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: combining patch-based and gradient-based approaches. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, 2003. 99

[31] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991. 16

[32] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael. Learning Low-Level Vision. *International Journal of Computer Vision*, 40(1):25–47, 2000. 36

[33] P. Gehler and M. Welling. Product of "edge-perts". *NIPS* 2005*. 50, 53

[34] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *PAMI*, 1984. 50, 61

[35] DM Greig, BT Porteous, and AH Seheult. Exact Maximum A Posteriori Estimation for Binary Images. *Journal of the Royal Statistical Society, Series B.*, 51(2):271–279, 1989. 73

[36] X. Han, C. Xu, and J.L. Prince. A topology preserving level set method for geometric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):755–768, 2003. 109

[37] KJ Hayworth, N. Kasthuri, R. Schalek, and JW Lichtman. Automating the collection of ultrathin serial sections for large volume TEM reconstructions. *Microscopy and Microanalysis*, 12(S02):86–87, 2006. 20, 115

[38] X. He, R Zemel, and M.C. Perpinan. Multiscale conditional random fields for image labeling. *CVPR 2004*. 50, 61, 68, 69

[39] S.W. Hell and J. Wichmann. Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy. *Optics Letters*, 19(11):780–782, 1994. 20

[40] M. Helmstaedter, K.L. Briggman, and W. Denk. 3D structural imaging of the brain with photons and electrons. *Current Opinion in Neurobiology*, 18(6):633–641, 2008. 20

[41] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 2006. 52, 74

[42] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006. 33, 127

[43] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. 34

[44] V. Jain, J.F. Murray, F. Roth, S. Turaga, V. Zhigulin, K.L. Briggman, M.N. Helmstaedter, W. Denk, and H.S. Seung. Supervised Learning of Image Restoration with Convolutional Networks. *ICCV 2007.* 22, 118

[45] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 769–776. 2009. 22

[46] E. Jurrus, M. Hardy, T. Tasdizen, P.T. Fletcher, P. Koshevoy, C.B. Chien, W. Denk, and R. Whitaker. Axon tracking in serial block-face scanning electron microscopy. *Medical Image Analysis*, 2008. 21

[47] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 1983. 78

[48] D.C. Knill and W. Richards. *Perception as Bayesian inference.* Cambridge Univ Pr, 1996. 16

[49] G. Knott, H. Marchman, D. Wall, and B. Lich. Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. *Journal of Neuroscience*, 28(12):2959, 2008. 20

[50] T.Y. Kong. Digital topology. *Foundations of Image Understanding*, pages 33–71. 87

[51] T.Y. Kong, AW Roscoe, and A. Rosenfeld. Concepts of digital topology. *Topology and its Applications*, 46(3):219–262, 1992. 87

[52] T.Y. Kong and A. Rosenfeld. Digital Topology: introduction and survey. *Computer Vision Graphics and Image Processing*, 48:357–393, 1989. 87, 88

[53] S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. *NIPS* 2004.* 40, 50, 61, 69

[54] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*, 2001. 40, 69

[55] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989. 77

[56] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 24, 30

[57] Y. LeCun, L. Bottou, G.B. Orr, and K.R. Muller. Efficient backprop. *Neural Networks: Tricks of the Trade*, 1998. 30, 31, 77

[58] Y. LeCun, F.J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004. 24, 30

[59] Yann LeCun and Fu Jie Huang. Loss functions for discriminative training of energy-based models. In *Proc. of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005. 40

[60] S.Z. Li. *Markov random field modeling in image analysis*. Springer-Verlag, New York, 2001. 36, 38, 43, 66, 68, 78

[61] J.W. Lichtman and J.R. Sanes. Ome sweet ome: what can the genome tell us about the connectome? *Current Opinion in Neurobiology*, 18(3):346–353, 2008. 20

[62] J. Livet, T.A. Weissman, H. Kang, R.W. Draft, J. Lu, R.A. Bennis, J.R. Sanes, and J.W. Lichtman. Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. *Nature*, 450(7166):56–62, 2007. 20

[63] S. Lyu and E.P. Simoncelli. Statistical modeling of images with fields of Gaussian scale mixtures. *NIPS* 2006*. 49, 53

[64] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge, 2003. 39

[65] J.H. Macke, N. Maack, R. Gupta, W. Denk, B. Sch
"olkopf, and A. Borst. Contour-propagation algorithms for semi-automated recon-

struction of neural processes. *Journal of Neuroscience Methods*, 167(2):349–357, 2008. 21

[66] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Proc. CVPR*, volume 1, 2008. 121

[67] J.F. Mangin, V. Frouin, I. Bloch, J. Régis, and J. López-Krahe. From 3D magnetic resonance images to structural representations of the cortex topography using topology preserving deformations. *Journal of Mathematical Imaging and Vision*, 5(4):297–318, 1995. 87

[68] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *ICCV*, 2001. 17, 53, 74, 101

[69] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, July 2001. 111

[70] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans Pattern Anal Mach Intell*, May 2004. 11, 17, 23, 97, 101, 102, 104, 106, 111

[71] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture. *NIPS*, 2003. 23, 101, 111, 112

[72] K.D. Micheva and S.J. Smith. Array tomography: a new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 55(1):25–36, 2007. 20

[73] Yuriy Mishchenko. Automation of 3d reconstruction of neural tissue from large volume of conventional serial section transmission electron micrographs. *J Neurosci Methods*, 176(2):276–289, Jan 2009. 21

[74] Joseph F. Murray and Kenneth Kreutz-Delgado. Visual recognition and inference using dynamic overcomplete sparse learning. *Neural Computation*, 19:2301–2352, 2007. 74

[75] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P.E. Barbano. Toward Automatic Phenotyping of Developing Embryos From Videos. *IEEE Trans. Image Proc.*, 2005. 24

[76] S. Parise and M. Welling. Learning in markov random fields: An empirical study. *Joint Statistical Meeting*, 2005. 39

[77] S. Parise and M. Welling. Learning in markov random fields: An empirical study. *Joint Stat. Meeting*, 2005. 50, 68, 74

[78] N. Passat, M. Couprie, and G. Bertrand. Minimal simple pairs in the 3-D cubic grid. *Journal of Mathematical Imaging and Vision*, 32(3):239–249, 2008. 88

[79] N. Passat and L. Mazo. An introduction to simple sets. *Pattern Recognition Letters*, 30(15):1366–1377, 2009. 88

[80] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *PAMI*, 12(7):629–639, 1990. 45, 65, 80

[81] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Image understanding*, 3:1–18, 1989. 16

[82] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Proc.*, 2003. 8, 18, 49, 53, 55, 58

[83] C. Pudney. Distance-ordered homotopic thinning: a skeletonization algorithm for 3D digital images. *Computer Vision and Image Understanding*, 72(3):404–413, 1998. 108

[84] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. *NIPS*, 17:2004, 2004. 69

[85] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971. 105

[86] M. Ranzato, YL Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. *NIPS* 2007*. 52, 127

[87] X. Ren. Multi-scale Improves Boundary Detection in Natural Images. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, pages 533–545. Springer, 2008. 111

[88] A. Rosenfeld. Continuous' functions on digital pictures. *Pattern Recognition Letters*, 4(3):184, 1986. 84, 85

[89] A. Rosenfeld, T.Y. Kong, and A. Nakamura. Topology-preserving deformations of two-valued digital pictures. *Graphical Models and Image Processing*, 60(1):24–34, 1998. 87, 88

[90] S. Roth. High-order markov random fields for low-level vision. PhD Thesis, Brown Univ., 2007. 36, 56

[91] S. Roth and M.J. Black. Fields of Experts: a framework for learning image priors. *CVPR 2005*. 8, 45, 46, 50, 53, 55, 56, 58, 61

[92] C. Rother, V. Kolmogorov, and A. Blake. " GrabCut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004. 126

[93] M.J. Rust, M. Bates, and X. Zhuang. Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (STORM). *Nature methods*, 3(10):793, 2006. 20

[94] F. Segonne. *Segmentation of medical images under topological constraints*. PhD thesis, Massachusetts Institute of Technology, 2005. 91, 93, 94, 95

[95] F. Segonne and B. Fischl. Integration of topological constraints in medical image segmentation. 81, 88

[96] H.S. Seung. Learning continuous attractors in recurrent networks. *NIPS* 1997*. 59

[97] H.S. Seung. Reading the Book of Memory: Sparse Sampling versus Dense Mapping of Connectomes. *Neuron*, 62(1):17–29, 2009. 20

[98] S. Smale, L. Rosasco, J. Bouvrie, A. Caponnetto, and T. Poggio. Mathematics of the neural response. *Foundations of Computational Mathematics*, pages 1–25, 2009. 128

[99] O. Sporns, G. Tononi, and R. Kotter. The human connectome: a structural description of the human brain. *PLoS Comput Biol*, 1(4):e42, 2005. 20

[100] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields. In *ECCV 2006*. 50

[101] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A Comparative Study of Energy Minimization Methods for Markov Random Fields. *Proc. Europ. Conf. Comp. Vision*, 2006. 38

[102] M.F. Tappen, C. Liu, E.H. Adelson, and W.T. Freeman. Learning Gaussian Conditional Random Fields for Low-Level Vision. *CVPR 2007*. 50, 53

[103] S.C. Turaga. *Ultrametric learning of image segmentation and hierarchies: with application to neural circuit reconstruction*. PhD thesis, Massachusetts Institute of Technology, 2009. 21, 106

[104] S.C. Turaga, J.F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H.S. Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, In Press. 99

[105] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):929–944, 2007. 96, 101, 105

[106] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995. 74

[107] L.R. Varshney, B.L. Chen, E. Paniagua, D.H. Hall, and D.B. Chklovskii. Structural Properties of the Caenorhabditis elegans Neuronal Network. *Arxiv preprint arXiv:0907.2373*, 2009. 20

[108] J. Weickert. Coherence-enhancing diffusion of colour images. *Image and Vision Computing*, 17(3):201–212, 1999. 80

[109] Y. Weiss and W.T. Freeman. What makes a good model of natural images? *CVPR 2007*. 8, 50, 53, 56

[110] M. Welling, G. Hinton, and S. Osindero. Learning sparse topographic representations with products of student-t distributions. *Advances in neural information processing systems*, pages 1383–1390, 2003. 45

[111] JG White, E. Southgate, JN Thomson, and S. Brenner. The Structure of the Nervous System of the Nematode Caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 314(1165):1–340, 1986. 20

[112] S.C. Zhu, Y. Wu, and D. Mumford. Filters, Random Fields and Maximum Entropy (FRAME): Towards a Unified Theory for Texture Modeling. *International Journal of Computer Vision*, 1998. 50