Chapter 10
Image Segmentation for Connectomics Using Machine Learning

T. Tasdizen, M. Seyedhosseini, T. Liu, C. Jones, and E. Jurrus

Abstract Reconstruction of neural circuits at the microscopic scale of individual neurons and synapses, also known as connectomics, is an important challenge for neuroscience. While an important motivation of connectomics is providing anatomical ground truth for neural circuit models, the ability to decipher neural wiring maps at the individual cell level is also important in studies of many neurodegenerative diseases. Reconstruction of a neural circuit at the individual neuron level requires the use of electron microscopy images due to their extremely high resolution. Computational challenges include pixel-by-pixel annotation of these images into classes such as cell membrane, mitochondria and synaptic vesicles and the segmentation of individual neurons. State-of-the-art image analysis solutions are still far from the accuracy and robustness of human vision and biologists are still limited to studying small neural circuits using mostly manual analysis. In this chapter, we describe our image analysis pipeline that makes use of novel supervised machine learning techniques to tackle this problem.

Introduction

Supervised machine learning techniques have shown great potential and have become important tools of choice in many problems. This is particularly true for image analysis [1, 2]. Image analysis approaches with hand designed, deterministic filters are being replaced with approaches that use filters and operations learned

T. Tasdizen (✉) • M. Seyedhosseini • C. Jones
Electrical and Computer Engineering Department, Scientific Computing and Imaging Institute, University of Utah, Lake City, UT 84112, USA
e-mail: tolga@sci.utah.edu

T. Liu • E. Jurrus
School of Computing, Scientific Computing and Imaging Institute, University of Utah, Lake City, UT 84112, USA

K. Suzuki (ed.), *Computational Intelligence in Biomedical Imaging*,
DOI 10.1007/978-1-4614-7245-2_10,

from human generated ground truth. This supervised learning strategy has been shown to outperform traditional methods in many image analysis applications. In this chapter, we will focus on one such application: neural circuit reconstruction from electron microscopy (EM) images at the scale of individual neurons and synapses. We will refer to this problem as connectomics [3–10].

An important motivation for connectomics is providing anatomical ground truth for neural circuit models. Connectomics is also important in studies of many neurodegenerative diseases. For instance, a loss of photoreceptors in the retina can cause neurons to rewire [11, 12] and neural circuits undergo remodeling in response to seizures in epilepsy [13, 14]. Serial section EM, where a block of tissue is cut into sections and imaged, has sufficient detail for identification of individual neurons and their synaptic connections in a three-dimensional (3D) volume; however, this is a difficult and time-consuming image analysis task for humans. Furthermore, state-of-the-art automated image analysis solutions are still far from the accuracy and robustness of human vision. Therefore, biologists are still limited to studying small neural circuits using mostly manual analysis. Reconstruction of a neural circuit from an electron microscopy volume requires segmentation of individual neurons in three dimensions and the detection of synapses between them. Supervised learning approaches to this problem typically involve pixel-by-pixel annotation of these images into classes such as cell membrane, mitochondria, and synaptic vesicles with a classifier learned from training data. We will refer to this step as the pixel classifier. This is generally followed by another step which segments individual neurons based on the cell membrane pixels detected by the classifier in the initial step. This second step can be as simple as a flood fill operation on the thresholded output of the pixel classifier from the first step or as complex as a second classifier which learns to merge/split regions obtained from the pixel classifier's output as in our approach. This two-step strategy is taken because cell membranes are similar in local appearance to many other intracellular structures (see Fig. 10.1) which makes their detection with deterministic filter banks or segmentation with techniques such as active contours very difficult.

The supervised learning strategy for connectomics has its own challenges that need to be addressed. First, generating training data from electron microscopy images can be a cumbersome task for humans. On the other hand, no training data is needed for deterministic approaches. Second, the training set can be extremely large since each pixel in the training image becomes a training example. This requires a lengthy training stage. In comparison, no training time is spent in deterministic approaches. Third, overfitting is a possibility as in any machine learning application. Finally, the cell membrane classification step demands extremely high accuracy. Even with high pixel accuracy rates such as 95 %, which is acceptable in many other applications, it is virtually certain that almost every neuron in a volume will be incorrectly segmented due to their global, tree-like structure, and correspondingly large surface area. The lack of reliable automated solutions to these problems is the current bottleneck in the field of connectomics. In this chapter, we will describe our approach to deal with each of these problems.

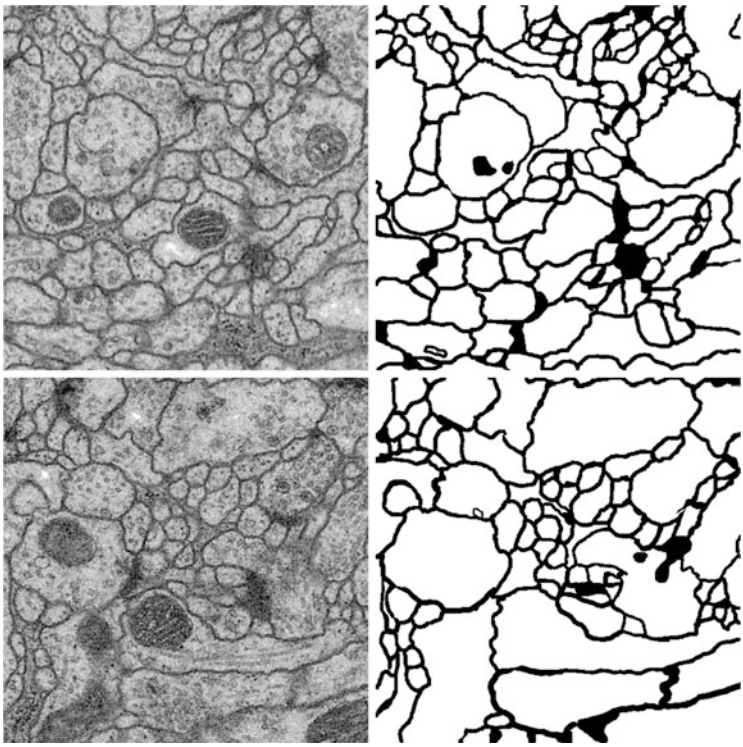

Fig. 10.1 *Left*: Two ssTEM sections from the Drosophila first instar larva VNC [8]. *Right*: Corresponding ground truth maps for cell membranes (*black*). *Data: Cardona Lab, ETH*

Background

Connectomics from Electron Microscopy

Compared with other imaging techniques, such as MRI [15] and scanning confocal light microscopy [16–20], electron microscopy provides much higher resolution and remains the primary tool for connectomics. The only complete reconstruction of a nervous system to date has been performed for the nematode *Caenorhabditis elegans* (*C. elegans*) which has 302 neurons and just over 6,000 synapses [21–23]. This reconstruction, performed manually, is reported to have taken more than a decade [4]. Recently, high throughput serial section transmission electron microscopy (ssTEM) [5, 6, 9, 24–26] and serial block-face scanning electron microscopy (SBFSEM) [4, 10, 27] have emerged as automated acquisition strategies for connectomics. Automatic Tape-Collecting Lathe Ultramicrotome (ATLUM) [28] is another promising technology for speeding up data collection for connectomics.

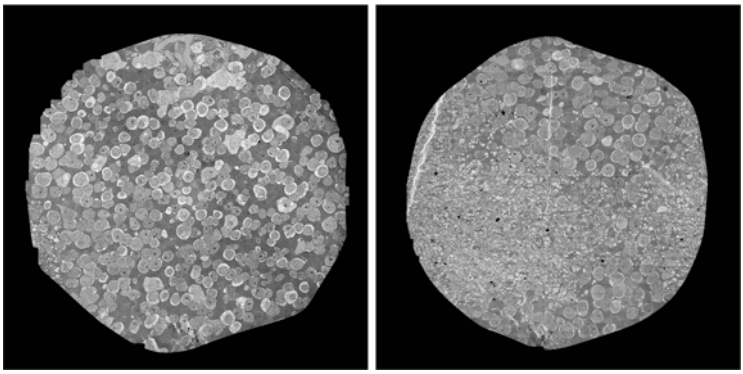

Fig. 10.2 Two sections from the retinal connectome [32] which comprises 341 sections. Each section is $\sim$ 32 GB and comprises 1,000 image tiles, each 4, 096 $\times$4, 096 pixels. The circular area with data is approximately 132, 000 pixels in diameter. The complete dataset is $\sim$ 16 TB. *Data: Marc Lab, University of Utah*

In the ssTEM technique, sections are cut from a specimen block and suspended so that an electron beam can pass through it, creating a projection which can be captured digitally or on film with 2 nm in-plane resolution. This extremely high resolution is sufficient for identifying synapses visually. An important trade-off occurs with respect to the section thickness: thinner sections, e.g. 30 nm, are easier to analyze because structures are crisper due to less averaging whereas thicker sections, e.g. 90 nm, are easier to handle physically without loss. Through mosaicking of many individual images [29, 30], ssTEM offers a relatively wide field of view to identify large sets of cells as they progress through the sections. Image registration techniques are necessary to align the sections into a 3D volume [6, 31].

In the SBFSEM technique, sections are cut away, and the electron beam is scanned over the remaining block face to produce electron backscatter images. Since the dimensions of the solid block remain relatively stable after sectioning, there is no need for image registration between sections. However, the in-plane resolution is closer to 10 nm which is a disadvantage compared to ssTEM. Typical section thicknesses for SBFSEM are 30–50 nm.

New projects using the techniques described above capture very large volumes containing several orders of magnitude more neurons than the *C. elegans*. As an example, Fig. 10.2 shows two mosaic sections from a *16 TB* ssTEM retina volume [32] that was assembled with our algorithms [6, 31]. It is not feasible to reconstruct complete neural circuits in these datasets with manual methods. Moreover, population or screening studies are unfeasible since fully manual segmentation and analysis would require years of manual effort per specimen. As a result, automation of the computational reconstruction process is critical for the study of these systems.

Finally, as an alternative to SBFSEM and ssTEM, fully 3D approaches such as focused ion-beam scanning electron microscopy (FIBSEM) [33] and electron tomography [34–36] produce nearly isotropic datasets. Both techniques are limited

to studying small volumes which is a disadvantage for their use for connectomics. FIBSEM uses a focused-ion beam to mill away sections instead of cutting sections with a knife. While it is clear that these datasets are easier to analyze accurately, the amount of data and the time it would take to acquire and analyze them is prohibitive for large-scale circuit reconstruction applications for the time being.

Neuron Segmentation

Figure 10.2b shows the complexity of the problem. This image which is 132, 000 pixels in diameter contains thousands of neuronal processes. The larger structures seen in Fig. 10.2a are the cell bodies of these processes. Reconstructing neural circuits from EM datasets involves segmenting individual neurons in 3D and finding the synapses between them. Neuron segmentation is the immediate challenge and thus has gathered significantly more attention than synapse detection. The only successful automatic synapse detection approaches so far have been limited to FIBSEM data which offers almost isotropic resolution [37]. In this chapter, we limit our attention to neuron segmentation. There are two general strategies for neuron segmentation. One strategy is to directly segment neurons in 3D [38, 39]. However, this can be difficult in many datasets due to the anisotropic nature of the data. The large section thickness often causes features to shift significantly between sequential images both in ssTEM and SBFSEM, decreasing the potential advantages of a direct 3D approach. The other strategy first segments neurons in two-dimensional (2D) images followed by linking them across sections to form a complete neuron [40–43]. Our approach fits in this second category. In this chapter, we focus on the 2D neuron segmentation problem. For linking 2D neuron regions across sections we refer the reader to [41, 44–46].

Image Processing Methods

Neuron segmentation has been studied mostly using semi-automated methods [8, 24, 40, 43, 47]. Fully automatic segmentation is complicated by two main challenges: complex intracellular structures such as vesicles and mitochondria that are present in the ssTEM images and the extremely anisotropic resolution of the data, e.g. 2 nm in-plane vs. 40 nm out-of-plane. Previous automatic EM segmentation methods include active contours which rely on an gradient term to drive the segmentation [42, 48–53]. This gradient/edge term can be ambiguous because of the locally similar appearance of neuron and intracellular membranes. Figure 10.1 shows two ssTEM sections from the Drosophila first instar larva ventral nerve cord (VNC) [8] and their corresponding cell membrane ground truth maps drawn by a human expert. Notice that the intensity profile of cell membranes completely overlaps with many other intracellular structures such as mitochondria (large, dark round structures) and synapses (elongated, dark structures). Furthermore, notice that when

the cell membranes are parallel to the cutting plane in 3D they appear fuzzy and of lighter intensity. In our earlier work, we used directional diffusion to attempt to remove intracellular structures from similar images [54]. Other researchers have used Radon-like features [55] to try to isolate cell membranes without using supervised learning. These deterministic methods have had limited success.

Furthermore, due to the very anisotropic resolution, a typical approach is to segment neurons in 2D sections followed by a separate stage to link the segments in 3D as mentioned earlier. Active contours can be propagated through the sections with the help of Kalman filtering [50]; however, this propagation can be inaccurate because of the large changes in shape and position of neurons from one section to the next. The large shape change stems from the anisotropy of the volume while the position change problem stems from the anisotropy as well as the fact that each section is cut and imaged independently resulting in nonrigid deformations. While our and other registration methods [31, 56, 57] can be used to fix the position change problem to a large extent, the shape change problem remains. Consequently, due to this poor initialization, active contours can get stuck on edges of intracellular structures and fail to segment neurons properly. Hence, active contours have been most successful in earlier SBFSEM images which only highlight extracellular spaces removing almost all contrast from intracellular structures. While this simplifies segmentation, it also removes important information such as synapses that are critical to identifying functional properties of neurons [7]. The other drawback is that active contours typically segment one neuron at a time whereas a typical volume has tens of thousands of neurons. While graph-cut methods [58–60] can simultaneously segment a large number of neurons, they still have the intracellular membrane problem. Combined with machine learning methods [61], they have an improved detection accuracy and can be used more reliably.

Machine Learning Methods

As discussed, intracellular structures are present in images which can be a source of confusion for neuron segmentation. Supervised lassifiers have been applied to the problem of neuron membrane detection as a precursor to segmentation and have proven more successful [5, 38, 39, 62, 63]. Membrane detection results can be with a method as simple as flood-filling for segmentation or as an edge term in active contour or graph-cut methods to overcome the problem due to intracellular structures. Jain et al. [39] use a convolutional network for restoring membranes in SBFSEM images. Similar to Markov random field (MRF) [64, 65] and conditional random fields (CRF) [66, 67] models, convolutional networks impose a spatial coherency on the membrane detection results. However, convolutional nets define a less rigid framework where the spatial structure is learned and they can make use of context from larger regions, but typically need many hidden layers. The large number of hidden layers can become problematic in training with backpropagation and simplifications such as layer-by-layer training are often needed [68]. The series

neural network architecture [63] used here also takes advantage of context and samples image pixels directly to learn membrane boundaries.

While supervised learning for cell membrane detection has met moderate success, all methods require substantial user interaction for initialization and correcting errors in the subsequent segmentation step [40]. As discussed in section "Introduction", the cell membrane classification step demands extremely high accuracy. Neurons have a global tree-like geometry with a correspondingly large surface area between neighboring neurons (cell membranes). A single local area of false negatives on this cell membrane leads to under-segmentation. Therefore, even with high pixel accuracy rates such as 95 % it is virtually certain that almost every neuron in a volume will be incorrectly under-segmented. Furthermore, neurons have very narrow cross-sections in many places which create many possibilities for over-segmentation when intracellular structures with similar local appearance to cell membranes are co-located with these constrictions. Researchers have investigated approaches to improve the accuracy of such classifiers. A 2-step classification where a membrane detection classifier is followed by a higher-level classifier that learns to remove spurious boundary segments causing over-segmentation was proposed [38]. Funke et al. [46] proposed a tree structure for simultaneous intra-section and inter-section segmentation. However, their model can only segment a 3D volume of consecutive sections and cannot segment a single section. Moreover, the final optimization problem in their model can be complicated given a set of complete trees of an image stack. Another promising direction is to optimize segmentation error rather than pixel-wise classification error, focusing learning on critical pixels where a mistake in classification results in a segmentation error [69, 70]. Topological constraints have also been proposed as an alternative to the pixel-wise classification error metric [71]. A recent study proposed to combine tomographic reconstruction with ssTEM to achieve a virtual resolution of 5 nm out-of-plane [72]. Finally, perceptual grouping applied to membrane detection classifier results was used in [61].

Another approach to improving the accuracy of cell membrane detection is to use multi-scale methods. In early computer vision work, the neocognitron [73], which is a network composed of alternating layers of simple cells for filtering and complex cells for pooling and downsampling inspired by Hubel and Wiesel [74], was proposed for object recognition. Learning in the neocognitron is unsupervised. Convolutional nets in their original form are similar to the neocognitron in terms of their architecture; however, learning is supervised [75]. Convolutional nets have been applied to face detection [76, 77], face recognition [78], and general object recognition [79, 80]. However, the convolutional nets applied to connectomics [39, 69–71] have not taken advantage of the multi-scale nature of the neocognitron. In a different microscopy imaging application, Ning et al. have applied multi-scale convolutional nets to the problem of segmentation of subcellular structures in Differential Interference Contrast microscopy [81]. We proposed a multi-scale version of our series neural network architecture [82] that we also employ here. Recently, deep convolutional nets have been proposed for learning hierarchical image features [83, 84]. While these deep convolutional nets are trained in an

unsupervised manner by presenting a set of training images containing the object of interest [85], their outputs can also be used as features in an object recognition application. This approach was recently used in the winning entry of the ISBI EM image segmentation challenge [86].

Methods

In this section, we will describe our algorithms for segmenting EM images. Section "Convolutional Networks and Auto-context Overview" discusses convolutional networks and auto-context methods which motivate our method. Section "Series of Artificial Neural Networks Pixel Classifier" introduced our series of artificial neural networks (ANN) framework. Section "Multi-scale Series of ANN Pixel Classifier" generalizes this framework to a multi-scale model. Section "Partial Differential Equation Based Post Processing" discusses a partial differential processing-based post-processing step to close gaps in the membrane detection results from the multi-scale series of artificial neural networks. This step typically results in a slight over-segmentation of the images. Therefore, section "Watershed Merge Tree Classifier" describes a watershed transform and supervised learning-based method for merging regions in the segmentation as necessary.

Convolutional Networks and Auto-Context Overview

As discussed in section "Machine Learning Methods", supervised machine learning methods have proven useful for detecting membranes in EM images. To address the challenges presented, we developed a machine learning method that combines two bodies of related work. The first by Jain et al. use a multilayer convolutional ANN to classify pixels as membrane or nonmembrane in specimens prepared with an extracellular stain [39]. The convolutional ANN has two important characteristics: it learns the filters for classification directly from data, and the multiple sequential convolutions throughout the layers of the network account for an increasing (indirect) filter support region. This method will work well for different types of image data, since it uses, as input, raw pixel data. In addition, the multiple convolutions enable the classifier to learn nonlocal structures that extend across the image without using large areas of the image as input. However, this method requires learning a very large number of parameters using backpropagation through many layers. Therefore, it is computationally intensive and requires very large training sets. Also of particular relevance is Tu's auto-context framework [87] from the computer vision literature, which uses a series of classifiers with contextual inputs to classify pixels in images. In Tu's method, the "continuous" output of a classifier, considered as a probability map, and the original set of features, are used as inputs

to the next classifier. The probability map values from the previous classifiers provide context for the current classifier, by using a feature set that consists of samples of the probability map at a large neighborhood around each pixel. Theoretically, the series of classifiers improves an approximation of an a posteriori distribution [87]. Hence, each subsequent classifier extends the support of the probability map, improving the decision boundary in feature space, and thus the system can learn the context, or shapes, associated with a pixel classification problem. Similar to the convolutional network, this means that a classifier can make use of information relayed by previous classifiers from pixel values beyond the scope of its neighborhood. However, the particular implementation demonstrated by Tu uses 8,000 nonspecific, spatially dispersed, image features, and a sampling of probability maps in very large neighborhoods. This is appropriate for smaller scale problems. On the other hand, for large connectomics datasets, it can be impractical to calculate thousands of features in order to train the classifier. Similar to Jain et al. we choose to learn the image features directly from the data and use the image intensities as input to our architecture, rather than preprocessing the data and computing thousands of image features. This provides us with a much smaller set of features and allows for flexibility and training of large datasets. Also, the use of the series ANNs and increasing context allows us to focus on small sets of image features to detect membranes, while also eliminating pixels that represent vesicles or other internal structures.

Series of Artificial Neural Networks Pixel Classifier

Problem Formulation

Let $X = (x(i, j))$ be a 2D input image that comes with a ground truth $Y = (y(i, j))$ where $y(i, j) \in \{-1, 1\}$ is the class label for pixel (i, j). The training set is $T = \{(X_k, Y_k); k = 1, \ldots, M\}$ where M denotes the number of training images. Given an input image X, the maximum a posteriori (MAP) estimation of Y for each pixel is given by

$$\hat{y}_{MAP}(i,j) = \operatorname{argmax} p(y(i,j)|X). \tag{10.1}$$

It is not practical to solve (10.1) for large real-world problems. Instead of the exact equation an approximation can be obtained by using the Markov assumption

$$\hat{y}_{MAP}(i,j) = \operatorname{argmax} p(y(i,j)|X_{N(i,j)}), \tag{10.2}$$

where $N(i, j)$ denotes all the pixels in the neighborhood of pixel (i, j). In practice, instead of using the entire input image, the classifier has access to a limited number of neighborhood pixels at each input pixel (i, j). This approximation decreases the computational complexity and makes the training tractable on large datasets.

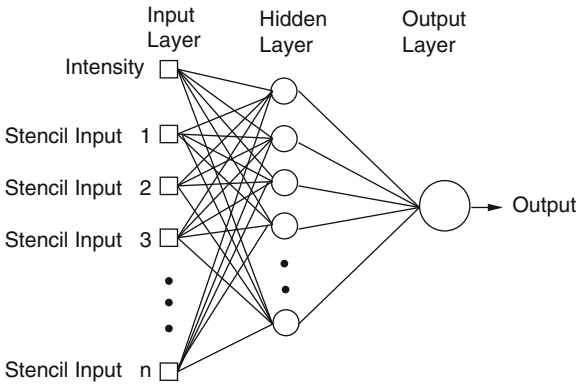

Fig. 10.3 Artificial neural network diagram with one hidden layer. Inputs to the network, in this framework, include the image intensity and the values of the image at stencil locations

Lets call the output image of this classifier $C = (c(i, j))$. In our series ANN, the next classifier is trained both on the neighborhood features of X and on the neighborhood features of C. The MAP estimation equation for this classifier can be written as

$$\hat{y}_{MAP}(i,j) = \text{argmax}\, p(y(i,j)|X_{N(i,j)}, C_{N'(i,j)}), \tag{10.3}$$

where $N'(i, j)$ denotes the neighborhood lattice of pixel (i, j) in the context image. Note that N and N' can represent different neighborhoods. The same procedure is repeated through the different stages of the series classifier until convergence. It is worth noting that (10.3) is closely related to the CRF model [66]; however, multiple models in series are learned which is an important difference from standard CRF approaches. It has been shown that this approach outperforms iterations with the same model [88].

Artificial Neural Network

Given the success of ANNs for membrane detection [5, 39], a multilayer perceptron (MLP) ANN is implemented as the classifier. An MLP is a feed-forward neural network which approximates a classification boundary with the use of nonlinearly weighted inputs. The architecture of the network is depicted schematically in Fig. 10.3. The output of each processing element (PE) (each node of the ANN) is given as [89, 90]

$$y = f(\mathbf{w}^T\mathbf{x} + b), \tag{10.4}$$

where f is, in this case, the *tanh* nonlinearity, $\mathbf{w}$ is the weight vector, and b is the bias. The input vector $\mathbf{x}$ to PEs in the hidden layer is the input feature vector discussed in more detail in section "Image Stencil Neighborhood". For the output PEs, $\mathbf{x}$ contains the outputs of the PEs in the hidden layer.

ANNs are a method for learning general functions from examples. They are well suited for problems without prior knowledge of the function to be approximated. They have been successfully applied to robotics [91, 92] and face and speech recognition [93, 94] and are robust to noise. Training uses gradient descent to solve for a solution which is guaranteed to find a local minimum. However, several trade-offs occur in training ANNs regarding the size of the network and the number of inputs. An ANN with too many hidden nodes can lead to overfitting of the network [89], resulting in a set of weights that fits well to the training data, but may not generalize well to test data. At the other extreme, if the number of hidden nodes is insufficient, the ANN does not have enough degrees of freedom to accurately approximate the decision boundary. The number of features should also be kept small to mitigate the problem of high dimensional spaces. Generally speaking, as the dimensionality of the input space increases, the number of observations becomes increasingly sparse which makes it difficult to accurately learn a decision boundary. Additionally, the training time tends to scale with the amount of training data and size of the network, and therefore training smaller networks with fewer features is generally preferable. Hence, the number of inputs to each ANN should be large enough to describe the data, but small enough for manageable training times.

Image Stencil Neighborhood

Good feature selection in any classification problem is critical. In this application, one possible approach uses large sets of statistical features as the input to a learning algorithm. These features can include simple local and nonlocal properties, including the pixel values, mean, gradient magnitude, standard deviation, and Hessian eigenvalues [38, 87, 95]. These attempt to present the learning algorithm with a large variety of mathematical descriptors to train on and are designed to work on a variety of data types. To achieve this generality, however, large numbers of these features are required to train a classifier. Another approach is to design a set of matched filters and apply them to an image to approximate a pixel's similarity to a membrane. This works well if the membranes in the image are uniform and respond well using cross correlation [96, 97]. Moreover, the design of the filter bank requires significant a priori knowledge of the problem. Yet, the fixed design may not be optimal for the dataset. Most importantly, the match filters have to be redesigned for datasets with different characteristics. On the other hand, learning these filters from training data, as in the case of convolutional networks [39], has the advantage that no a priori knowledge is required. A similar idea has been used in texture classification where it was shown that direct sampling of the image with a patch is actually a simpler and more universal approach for training a classifier compared to the use of filter banks [98]. Image patches have also been used successfully for texture segmentation [99] and image filtering [100–102]. Similarly, using image neighborhoods as in (10.2) allows the ANNs to learn directly on the input intensity data, giving the classifier more flexibility in finding the correct

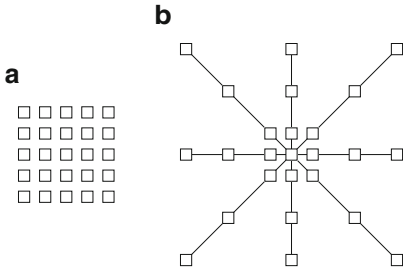

Fig. 10.4 Two image neighborhood sampling techniques: image pixels sampled using (**a**) a patch and (**b**) a stencil. For this example, the stencil contains the same number of samples, yet covers a larger area of the data. This is a more efficient representation for sampling the image space.

decision boundary. A square image neighborhood can be defined as an image patch, shown in Fig. 10.4a, centered at pixel k, l,

$$N(i,j) = \left\{ I_{k+i,l+j} : k,l = -\frac{R-1}{2}, \ldots, \frac{R-1}{2} \right\}. \tag{10.5}$$

R is the width of the square image patch. Unfortunately, the size of the image patches required to capture sufficient context can be quite large. For this reason, we propose using as input to the ANNs the values from the image and probability map of the previous classifier sampled through a stencil neighborhood, shown in Fig. 10.4b. A stencil is also centered at pixel k, l and defined as,

$$N(i,j) = \cup_{a=1}^{n} B(i,j,a) \tag{10.6}$$

where

$$B(i,j,a) = \left\{ I_{i+ak,j+al} : k,l = -1,0,1 \right\}, \tag{10.7}$$

and n is the number of rows the stencil spans in the image. The stencil in Figs. 10.4 and 10.5 cover large areas representing the desired feature space, but samples it in a spatially adaptive resolution strategy. For large image features, stencils such as the one in Fig. 10.5 are required. In this way, an ANN can be trained using a low-dimensional feature vector from image data, without having to use the whole image patch. Since the number of weights to be computed in an ANN is dominated by the connection between the input and the hidden layers, reducing the number of inputs reduces the number of weights and helps regularize the learned network. Moreover, using fewer inputs generally allows for faster training. With this, one aims to provide the classifier with sparse, but sufficient context information and achieve faster training, while obtaining a larger context which can lead to improvements in membrane detection. This strategy combined with the serial use of ANNs grows the region of interest for classification within a smaller number of stages and without long training times.

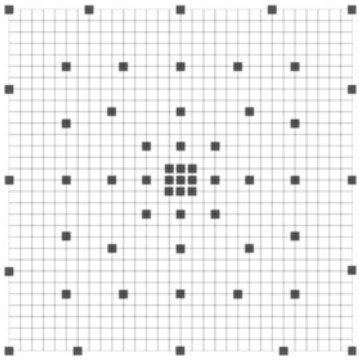

Fig. 10.5 Example of a larger image neighborhood sampling technique, covering a 31 ×31 patch of image pixels

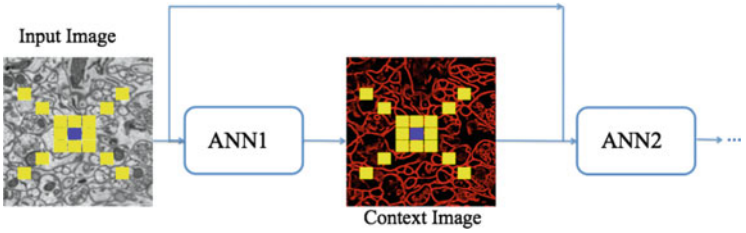

Fig. 10.6 Series neural network diagram demonstrating the flow of data between ANNs. The *blue* and *yellow squares* symbolize the center pixel and its neighborhood pixels in the stencil structure, respectively

Series Artificial Neural Networks

From the principles from auto-context, we architect a series of classifiers that leverage the output from previous networks to gain knowledge of a large neighborhood. The input to the first classifier is the image intensities around a pixel sampled using a stencil as described in section "Image Stencil Neighborhood". For the ANNs in the remaining series, the input vector contains the samples from the original image, used as input to the first ANN, appended with the values from the output of the previous classifier which was also sampled through the stencil neighborhood, yielding a larger feature vector. This second classifier is described mathematically with (10.3). While the desired output labels remain the same, each ANN is dependent on the information from the previous network and therefore must be trained sequentially, rather than in parallel. Figure 10.6 demonstrates this flow of data between classifiers. The lattice of squares represent the sampling stencil (shown more precisely in Fig. 10.5).

In summary, the series structure allows the classifiers to gather, with each step, context information from a progressively larger image neighborhood to the pixel being classified, as occurs with a convolutional ANN. The pixel values are sampled with a stencil neighborhood over each pixel, containing the pixels within the stencil (Fig. 10.5). The probability map feature vector is also obtained with a stencil

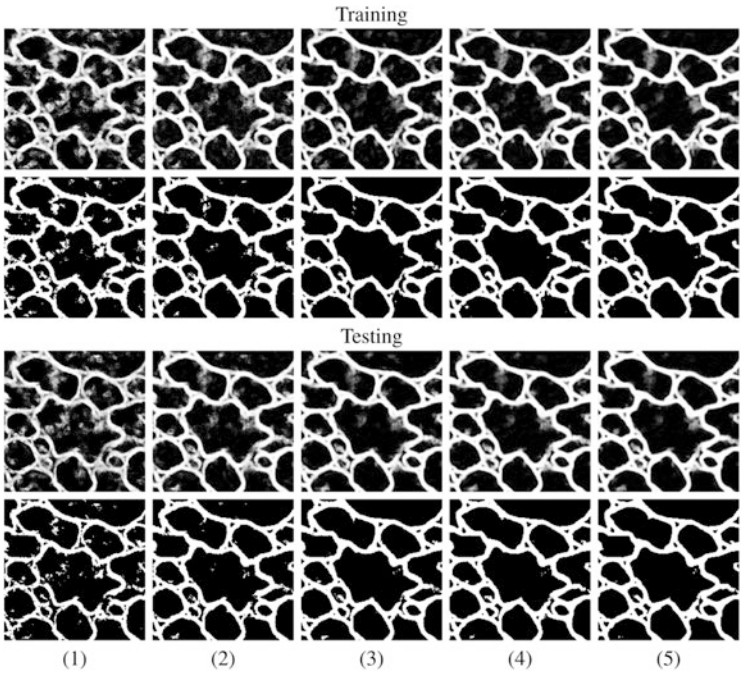

Fig. 10.7 Example output using the same image, first as part of a training set (*top two rows*), and then separately, as part of a testing set (*bottom two rows*), at each stage (1–5) of the network series. The output from each network is shown in rows 1 and 3. Rows 2 and 4 demonstrate the actual membrane detection when that output is thresholded. The network quickly learns which pixels belong to the membranes within the first 2–3 stages and then closes gaps in the last couple of stages

neighborhood placed over each pixel containing information about the classes, as determined by the previous classifier. Indirectly, the classification from the previous ANN contains information about features in surrounding pixels, that is not represented in the original feature set. This allows the subsequent networks in the series (Fig. 10.6) to make decisions about the membrane classification utilizing nonlocal information. Put differently, each stage in the series accounts for larger structures in the data, taking advantage of results from all the previous networks. This results in membrane detection that improves after each network in the series. Figure 10.7 visually demonstrates the classification improving between ANNs in the series as gaps in weak membranes are closed and intracellular structures are removed with each iteration in the series. The receiver operating characteristic (ROC) curves in Fig. 10.8 also demonstrate the increase in detection accuracy after each ANN in the series. Notice that the results converge after a few stages.

Combining the original image features with features sampled from the output of the previous classifier is important because, in this way, the membrane structure relevant for detection is enforced locally and then again at a higher level with each step in the series of classifiers. One of the advantages of this approach is that it

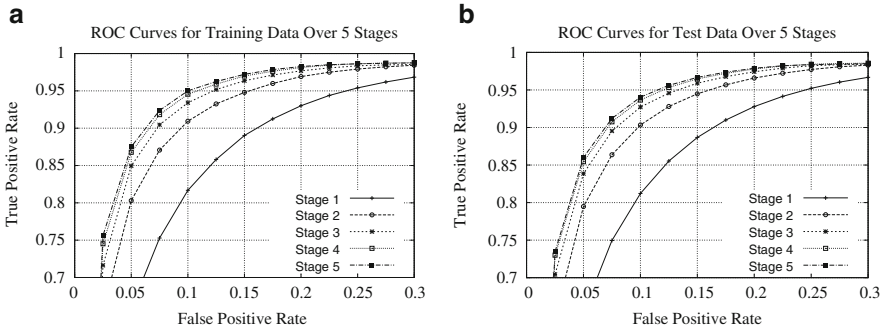

Fig. 10.8 ROC curves for the (**a**) training data and (**b**) testing data for each stage of the network on the *C. elegans* dataset

provides better control of the training, allowing the network to learn in steps, refining the classification at each step as the context information it needs to correctly segment the image increases. Again, note that the membrane structure is learned directly from the data. Compared to a single large network with many hidden layers and nodes, such as the convolutional ANN of Jain et al. [39], the proposed classifier is easier to train. This is mainly because each of the ANNs has a relatively small number of parameters. For example, for a single ANN, the number of parameters needed is approximately 500 for the first ANN and 1,100 for the remaining ANNs in the series. The number of weights in an ANN with a single-hidden layer is given by $(n + 1)h + (h + 1)$, where n is the number of inputs and h is the number of nodes in the hidden layer. For the first ANN in the series, $n = s$, where s is the number of points in the stencil. For the remaining ANNs in the series, $n = 2s$, since the original image and the output from the previous classifier are each sampled once. The total number of parameters across the whole series totals to approximately 5,000. In contrast, a convolutional ANN needs $(n + 1)h$ for the first layer, and $(n\,h + 1)h$ for the remaining layers, an h^2 dependence [39]. Hence, much less training data is needed in this approach, which is hard to obtain, since the ground truth must be hand labeled.[1] Furthermore, the training is simpler since backpropagation is less likely to get stuck on local minima of the performance surface [89, 90], and the network will train much faster. Moreover, this accounts for a smaller and simpler network which can be trained from smaller numbers of features in the input vector. The series of ANNs is much more attractive to train, as opposed to using a single large network with many hidden layers and nodes. A single large network would be time consuming and difficult to train due to the many local minima in the performance surface.

[1] According to the "rule-of-thumb" in [90], one needs at least 10 ×training samples of the total number of parameters. Thus, compared to Jain et al. [39] convolutional ANN, the approach presented here needs about 27 ×less training samples, for the values given.

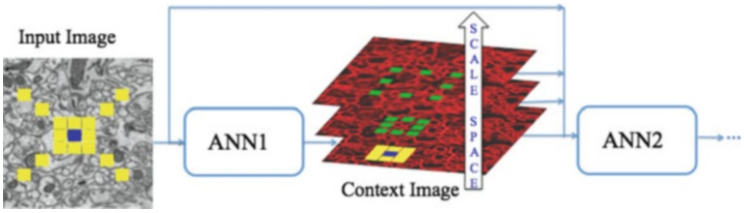

Fig. 10.9 Illustration of the multi-scale contextual model. Each context image is sampled at different scales (*green squares*). The *blue squares* represent the center pixel and the *yellow squares* show the selected input/context image locations at original scale

Multi-scale Series of ANN Pixel Classifier

In this section, we discuss how more information can be obtained by using a scale-space representation of the context and allowing the classifier access to samples of context at different scales. It can be seen from (10.3) that context image provides prior information to solve the MAP problem. Although the Markov assumption is reasonable and makes the problem tractable, it still results in a significant loss of information from global context because it only uses local information obtained from the neighborhood area. However, it is not practical to sample every pixel in a very large neighborhood area of the context due to computational complexity problem and overfitting. The series classifiers exploit a sparse sampling approach to cover large context areas as shown in Fig. 10.5. However, single pixel contextual information in the finest scale conveys only partial information about its neighborhood pixels in a sparse sampling strategy while each pixel in the coarser scales contains more information about its neighborhood area due to the use of averaging filters. Furthermore, single pixel context can be noisy whereas context at coarser scales is more robust against noise due to the averaging effect. In other words, while it is reasonable to sample context at the finest level a few pixels away, sampling context at the finest scale tens to hundreds of pixels away is error prone and results in a non-optimal summary of its local area. We will show how more information can be obtained by creating a scale space representation of the context and allowing the classifier access to samples of small patches at each scale. Conceptually, sampling from the scale space representation increases the effective size of the neighborhood while keeping the number of samples small.

Multi-scale Contextual Model

The multi-scale contextual model is shown in Fig. 10.9. In the conventional series structure, the classifiers simply take sparsely sampled context together with input image as input. In the multi-scale contextual model, each context image is treated as an image and a scale-space representation of context image is created by applying a

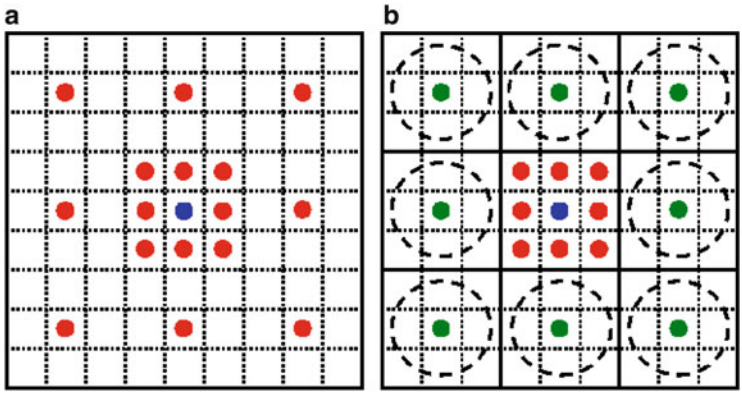

Fig. 10.10 Sampling strategy of context: (**a**) Sampling at a single scale (**b**) sampling at multi-scale. *Green circles* illustrate the summary of pixels in *dashed circles*. We use linear averaging to create the summary

set of averaging filters. This results in a feature map with lower resolution that is robust against the small variations in the location of features as well as noise. Figure 10.10 shows the multi-scale sampling strategy versus the single-scale sampling strategy. In Fig. 10.10b the classifier can have as an input the center 3×3 patch at the original scale and a summary of eight surrounding 3×3 patches at a coarser scale (The green circles denote the summaries of dashed circles). The green circles in Fig. 10.10b are more informative and less noisy compared to their equivalent red circles in Fig. 10.10a. The summaries become more informative as the number of scales increases. For example, in the first scale the summary is computed over 9 pixels (3×3 neighborhood) while it is computed over 25 pixels (5×5 neighborhood) in the second scale. Different methods such as Gaussian filtering, maximum pooling, etc. can be used to create the summary (green dots in Fig. 10.9) . From a mathematical point of view, (10.3) can be rewritten as:

$$\hat{y}_{MAP}(i,j) = \text{argmax}\, p(y(i,j)|X_{N(i,j)}, C_{N'_0(i,j)}(0), C_{N'_1(i,j)}(1), \ldots, C_{N'_l(i,j)}(l)) \quad (10.8)$$

where $C(0), C(1), \ldots, C(l)$ denote the scale space representation of the context and $N'_0(i,j), N'_1(i,j), \ldots, N'_l(i,j)$ are corresponding neighborhood structures. Unlike (10.3) that uses the context in a single-scale architecture, (10.8) takes advantage of multi-scale contextual information. Even though the Markov assumption is still used in (10.8), the size of the neighborhood is larger and thus less information is lost compared to (10.3).

The series multi-scale contextual model updates the (10.8) iteratively:

$$\hat{y}^{k+1}_{MAP}(i,j) = \text{argmax}\, p(y(i,j)|X_{N(i,j)}, C^k_{N'_0(i,j)}(0), C^k_{N'_1(i,j)}(1), \ldots, C^k_{N'_l(i,j)}(l)) \quad (10.9)$$

where $C^k(0), C^k(1), \ldots, C^k(l)$ are the scale space representation of the output of classifier at stage $k, k = 1, \ldots, K-1$ and $\hat{y}_{MAP}^{k+1}(i,j)$ denotes the output of the stage $k+1$. In turn, the $k+1$'th classifier output as defined in (10.9) creates the context for the $k+2$'th classifier. The model repeats (10.9) until the performance improvement between two consecutive stages becomes small. Because context is being used more effectively, the performance improvement through the stages is larger compared to the conventional series-ANN algorithm.

The overall performance of the multi-scale contextual model can be improved by extracting powerful features from the input image in addition to pixel intensities. It has been shown empirically that trying to segment the structures in connectome images using only geometric or textural features is not very effective [55]. Radon-like features (RLF) were proposed as a remedy to this problem as they are designed to leverage both the texture and the geometric information present in the connectome images to segment structures of interest. We refer the reader to [55] for further details of the RLF method. RLF method is an unsupervised method by itself but it can be integrated into supervised models as a feature extraction step. Furthermore, more powerful features can be obtained by computing RLF at multiple scales and for different edge threshold settings [82]. This richer set of features allow for correct detection of cell boundaries in the regions that cannot be detected by the original RLF as proposed in [55] and avoids the need for extensive parameter tuning.

Partial Differential Equation-Based Post-Processing

The partial differential equation (PDE) post-processing step is an entirely unsupervised process that improves the probability map by closing small to medium sized gaps in the membrane detection results. Typically the PDE post processing will generate an over-segmented image. Our motivation is that we can learn to fix over-segmentation errors with the watershed merge tree classifier as will be discussed in section "Watershed Merge Tree Classifier" whereas this is not possible for under-segmentation errors. In this section, we discuss how the probability map is updated at each iteration and the influence each term in the update equation has on the result.

Let f be an probability map where 1 represents locations with a low probability of being cell membrane and 0 represents locations with a high probability of being cell membrane. F is updated according to the rule $f_{k+1} = f_k + \Delta t(\Delta f)$, where

$$\Delta f = \alpha \Delta A + \beta \Delta B + \eta \Delta C \tag{10.10}$$

and each term $\Delta A, \Delta B$, and ΔC represents a different characteristic of the probability map or underlying image that is to be optimized. The $\Delta\ t$ term is a parameter that can be adjusted to improve the stability of the update rule and the α, β, and η terms

are parameters that can be used to control how much weight each different term in the PDE has relative to the other terms.

The first term in (10.10), ΔA, is defined as:

$$\Delta A = |\nabla f| \nabla \cdot \frac{\nabla f}{|\nabla f|} \tag{10.11}$$

where ∇ is the gradient operator and $\nabla \cdot$ is the divergence operator. The $\nabla \cdot \frac{\nabla f}{|\nabla f|}$ term in (10.11) computes the mean curvature at each pixel location in f and multiplying by $|\nabla f|$ ensures the stability. The effect of using the curvature is to force some smoothness along the boundaries between the membrane and non-membrane regions. Because the cells are generally large rounded structures with few sharp corners, high curvature areas are uncommon resulting in the curvature minimization term having the effect of favoring objects shaped like the interior of a typical cell. Without any other terms however, this would eventually reduce the areas with values close to 1 to shrink down to a small circle and eventually a single point. The other terms will counteract this behavior to give the desired result. In the discrete implementation of this curvature term, finite central difference is used to compute ∇f.

The second term in (10.10), ΔB, is defined as:

$$\Delta B = \nabla f \cdot \nabla G \tag{10.12}$$

where $\nabla G = \exp(-\frac{|\nabla I|^2}{\sigma})$ and I is a version of the original image filtered with the nonlocal means algorithm [101] to reduce the effects of noise. We chose the nonlocal means algorithm due to its success with textured images. The intent of this term is to push the edges of the probability map f to be along the edges of the original image. We assume that there is a strong edge between membrane and non-membrane regions. By itself this would produce a very jagged edge because of the noisy nature of the image. Combined with the curvature term, the edges of the probability map f will produce a clean edge that closely follows the edges in the original image. In the discrete implementation of this gradient term, an upwind scheme is used to compute ∇f and finite central difference is used to compute the ∇G and ∇I.

The final term in (10.10), ΔC, is defined as:

$$\Delta C = -.5\lambda_1 + .5\lambda_2 \tag{10.13}$$

where λ_1, λ_2 are the eigenvalues of the Hessian matrix of f, and $\lambda_1 > \lambda_2$. The larger eigenvalue of the Hessian of f represents the change across the gradient of f. By subtracting this term, it has the effect of inverse diffusion which tends to sharpen image features. This will effectively bring together areas of wide gray into a narrow dark region. Without the curvature term and gradient term this will cause some spurious detail to form, so a balance between this term and the other two terms is

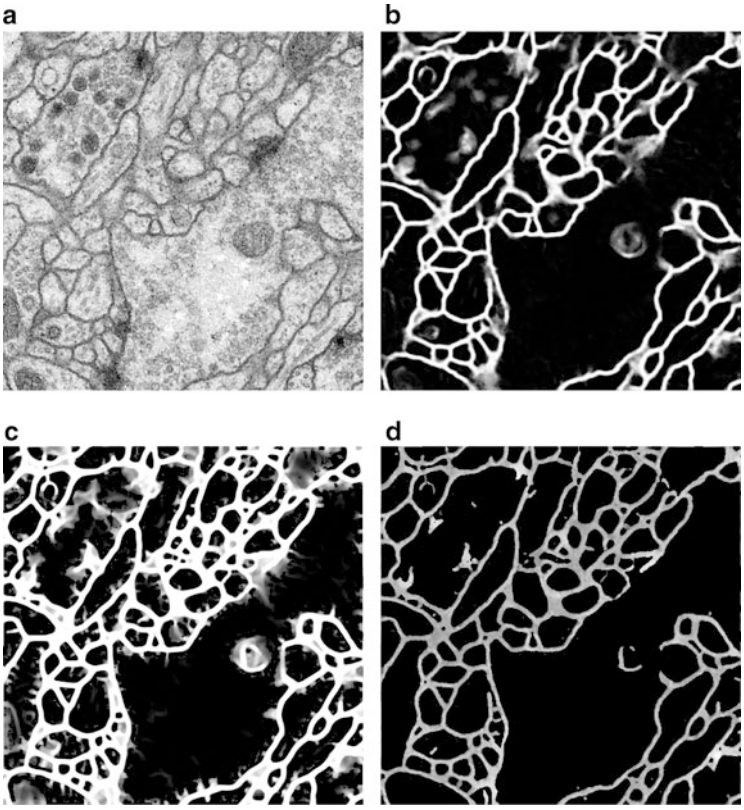

Fig. 10.11 Example of (**a**) original EM image, (**b**) multi-scale contextual membrane detection, (**c**) initial PDE result prior to thresholding , and (**d**) PDE result with replacement

necessary to ensure stability. The smaller eigenvalue of the Hessian of f represents the change in the direction perpendicular to the direction of the maximum gradient. Adding this term in allows growth extending the membranes at terminal points and connecting across regions that were missed in the initial probability map. In the discrete implementation of the Hessian, central differences are used to compute each of the 2nd derivatives. The number of iterations for the PDE is determined empirically according to the number of iterations that give the minimum rand error on the training data used in previous stages.

The result of running this algorithm is that the threshold giving the best result is at 1 resulting in everything less than 1 being considered non-membrane and everything equal to 1 being considered as membrane as seen in Fig. 10.11c. On some datasets this still offers significant improvement over just using the multi-scale series ANN; however, on other datasets the improvement is minimal. To be able to improve the thresholding and to be able to run the watershedding method described in the next section, we threshold the results of this PDE at the optimal

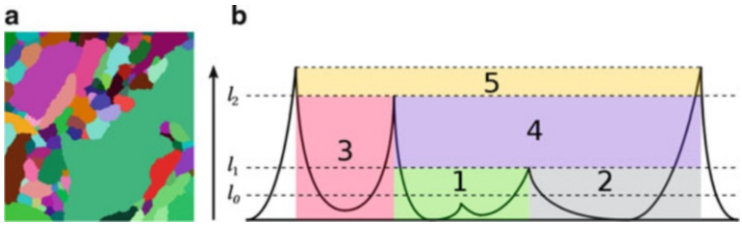

Fig. 10.12 Example of (**a**) initial watershed segmentation and (**b**) region merging with water level rising

threshold as determined on the training data and replace the areas classified as membrane with the intensity values from the original image. The result is an image where everything considered to be non-membrane has a value of 0 and everything considered to be membrane has the values from the original image as seen in Fig. 10.11d. Using the ISBI dataset, this algorithm improved both the training and the testing rand error as compared to the multi-scale series ANN by over 7 %.

Watershed Merge Tree Classifier

With the membrane detection and the PDE-based post-processing, we have a probability map for each image section indicating how probable every pixel can be membrane, as shown in Fig. 10.11. By simply applying thresholding on this map, we are able to get a segmentation. With this method, however, small mispredictions in pixels about membrane could lead to undesirable region merging and thus significant under-segmentation errors. With the watershed algorithm, we can also obtain a set of different segmentations by varying the water level. Yet a fixed global water level that works well through the entire image is difficult or even impossible to find due to various local terrains. On the contrary, we expect better results if we make specific local decisions according to different local situations, which motivates our watershed merge tree-based method.

Watershed Merge Tree

Consider a probability map as a three-dimensional terrain map with pixel probability as ridge height. Water rains into catchment basins, and regions with lower heights are flooded. An initial water level forms an initial segmentation as shown in Fig. 10.12a. With more water falling in and the water level rising over ridges, small regions merge into larger ones, and finally into one large region once the water level rises above the highest ridge in the map. Figure 10.12b gives a one-dimensional case for illustration: with initial water level l_0, we have regions 1, 2, and 3; when the

water level rises to l_1, regions 1 and 2 merge to 4; region 3 merges with 4 and 5 at water level l_2. This technique produces a hierarchy of region merging that can be represented by a tree structure, which we call a watershed merge tree.

Here we give a formal definition of a watershed merge tree. A watershed tree $T = (\{N\}, \{E\})$, derived from the concept of a tree in the graph theory, consists of a set of nodes and edges between them. At depth d, a node N_i^d corresponds to an image region R_i^d; an edge from a parent node N_i^d to its child node $N_{i'}^{d+1}$ indicates region $R_{i'}^{d+1}$ is a subregion of region R_i^d; a local tree structure $(N_i^d, N_{i'_1}^{d+1}, N_{i'_2}^{d+1}, \ldots)$ represents region R_i^d can be the merging result of all of its subregion $\{R_{i'_1}^{d+1}, R_{i'_2}^{d+1}, \ldots\}$. For simplicity, we here consider the merge tree as a binary tree, which means only two regions merge each time. If several regions merge at the same water level, we merge two regions at a time and the merging order can be arbitrary.

As we use nonlocal features for the boundary classifier, which will be described in detail in the next section, it is difficult to extract some meaningful features from regions that are too small. Therefore, we use an initial water level l_0 to merge some very small regions beforehand in the initial segmentation, and further conduct a preprocessing step to get rid of regions smaller than n_r pixels by merging them with their neighbor regions with the lowest probability barrier.

Boundary Classifier

In order to decide which regions we should preserve as the final segmentation in the merge tree, we need information about how probable each potential merge could happen. Thus, we learn a boundary classifier from training data. For a pair of regions, we consider the set of pixels that are adjacent to the other region as a boundary. The output of the classifier is a probability that the boundary between the two regions is false, or in other words, the two regions should merge. The input of the classifier is a set of 141 features extracted from the two merging regions, including geometric features (region area, boundary lengths, region contour lengths, etc.), intensity statistics features of boundary pixels from both original EM images and membrane detection probability maps, and regional features (texton histogram difference and watershed region merging saliency). Here the watershed region merging saliency is defined as the difference between the minimum water level to merge the two regions and the minimum value in the membrane detection probability map.

We obtain the label that indicates whether a region pair (R_i^d, R_j^d) should merge or not by measuring the Rand error over the ground truth segmentation, which is defined as

$$E_k = \frac{1}{|R_i^d| \cdot |R_j^d|} \sum_{x_p, x_q \in R_i^d \cup R_j^d} \left| \sigma_{pq} - \beta_{pq}^k \right|, \tag{10.14}$$

where (x_p, x_q) represents any pixel pair from the union of the two merging regions, and

$$\begin{aligned}
\sigma_{pq} &= \begin{cases} 1 & \text{if } xp \text{ and } xq \text{ are in the same truth region} \\ 0 & \text{otherwise} \end{cases} \\
\beta^1_{pq} &= \begin{cases} 1 & \text{always} \\ 0 & \text{never} \end{cases} \\
\beta^2_{pq} &= \begin{cases} 1 & \text{if } xp \text{ and } xq \text{ are in the same merging region} \\ 0 & otherwise. \end{cases}
\end{aligned} \tag{10.15}$$

Here the Rand error E_1 measures the portion of pixel pairs that are misclassified against the ground truth segmentation if the two regions merge, and E_2 is that portion if the two regions keep split. Thus, we can decide the label by simply comparing the Rand errors as

$$l^d_{ij} = \begin{cases} +1(merge) & \text{if } E^1_R < E^2_R \\ -1(split) & \text{otherwise.} \end{cases} \tag{10.16}$$

To balance the contributions of positive and negative examples, different weights are assigned to each type of examples. A random forest classifier [103] is trained with the weighted training examples and applied to make predictions about how likely a pair of regions should merge for the testing data.

Resolving the Merge Tree

The boundary classifier predicts the probability for every potential merge in a merge tree. We seek to take advantage of this information and obtain a consistent segmentation of the whole image in a sense of optimization. We define the consistency as that in the final segmentation any pixel should be labeled exactly once. In the context of our tree structure, if a node is selected, all of its ancestor and descendants cannot be selected, and its immediate sibling or a set of the descendants of its immediate sibling must be selected; if a node is not selected, one of its ancestors or a set of its descendants must be selected. In other words, exactly one node should be picked on each path from any leaf to the root. Figure 10.13 shows an artificial example. We have an initial over-segmentation shown in Fig. 10.13a, from which a merge tree is built as shown in Fig. 10.13c. Nodes 3, 6, 9, and 12 are picked for a consistent final segmentation shown in Fig. 10.13b. Consequently, the other nodes cannot be selected. Because we cannot have both purple region (node 9) and region 1 (or 2) exist, otherwise region 1 (or 2) would be labeled more than once as 1 (or 2) and 9, which is inconsistent by our definition. Meanwhile, if we select node 3, node 9 or nodes 1 and 2 together then must be picked in this case, otherwise

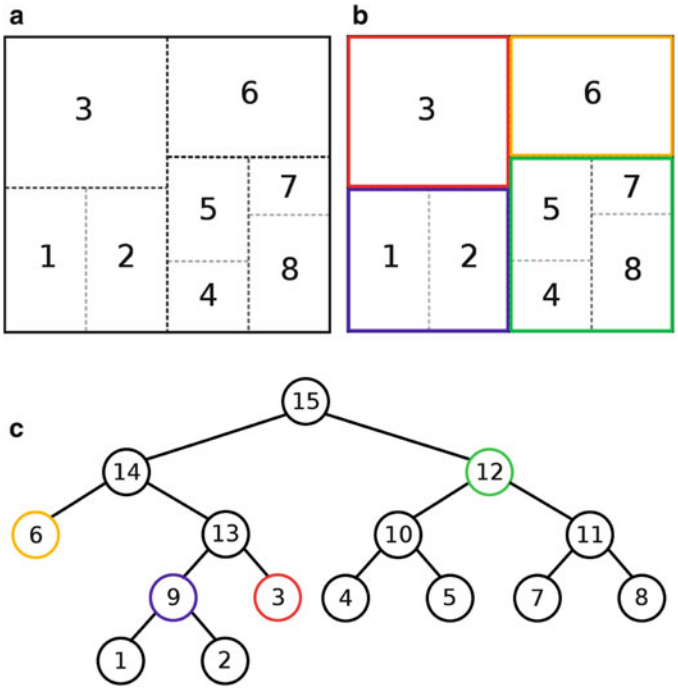

Fig. 10.13 Example of (**a**) initial over-segmentation, (**b**) consistent final segmentation, and (**c**) corresponding merge tree

region 1 or region 2 is not labeled in the final segmentation, which is also inconsistent.

In order to resolve the merge tree, we transform the probabilities of region pair merging into the form of potentials for each node in the tree. A region exists in the final segmentation because it neither splits into smaller regions nor merges with others into a larger region. Since each prediction that the classifier makes depends only on the two merging regions, we compute the potential that a node N_i^d is picked as the probability that its two child nodes $N_{i_1'}^{d+1}$ and $N_{i_2'}^{d+1}$ merge and at the same time N_i^d does not merge with its immediate sibling node N_j^d at the next higher water level to their parent node N_k^{d-1}. Thus, we define the potential for N_i^d as

$$P_i^d = p_{i_1',i_2'}^{d+1} \cdot (1 - p_{i,j}^d), \tag{10.17}$$

where $p_{i_1',i_2'}^{d+1}$ is the predicted probability that the two child nodes $N_{i_1'}^{d+1}$ and $N_{i_2'}^{d+1}$ merge, and $p_{i,j}^d$ is the probability that node N_i^d merge with its immediate sibling node N_j^d. In the example shown in Fig. 10.13c, the potential of node 9 is $P_9 = p_{1,2}(1 - p_{3,9})$. Since leaf nodes have no children, their potentials are defined as the

probability that they do not merge penalized into half. Similarly, the root node has no parent, so its potential is half of the probability that its children merge.

Given the potentials of each node, we seek to locally optimize the node selection to form a complete consistent final segmentation. Here we apply a greedy approach. The node with the highest potential in the merge tree is picked. Then all of its ancestors and descendants are regarded as inconsistent options and removed from the tree. This procedure is repeated until there are no nodes left in the tree. All the picked nodes together make up a complete consistent final segmentation.

Results

In this section, we will demonstrate the results of our algorithms on two ssTEM and one SBFSEM dataset. In addition to visual results, quantitative results are provided using the pixel error and rand error metrics on the training and testing datasets.

C. elegans *Ventral Nerve Cord ssTEM*

Dataset

The nematode *C. elegans* is an important dataset for neural circuit reconstruction. Despite being a well-studied organism [21], there are still numerous open questions such as how genes regulate wiring [104] or how connectivity is altered to mediate different behaviors, for example between males and females [105]. Reconstructions of the full nervous system reveal topological characteristics important for researchers studying neuron wiring. The particular ssTEM dataset used here is from the VNC of the *C. elegans* and is important for studying the topological structure resulting from neurons making connections to local targets.

Series-ANN Pixel Classifier

In this experiment, a series classifier with 5 stages was trained. Additional networks could be included; however, for these datasets, the performance converges to a limit (Fig. 10.8) and improvement in membrane detection is minimal. Each ANN used in the experiments contained one hidden layer with 20 nodes. We experimented with more layers and different numbers of nodes but did not find significant advantages. It is important that the number of nodes be large enough to approximate a nonlinear boundary and small enough that the ANN does not overfit to the training data [106, 107]. Results using 10, 20, and 30 nodes turned out to be somewhat similar. Given the time versus performance trade-off, we chose 20 nodes. The networks were trained using backpropagation with a step size of 0. 0001 and

momentum term of 0.5. We used early stopping as the criterion to determine when to terminate training [89, 90]. This means that a small portion of the training data (20 % in our case), called the validation set, is used only to test the classifier generalization performance. The training terminates when the lowest error on the validation set is attained. To mitigate problems with local minima, each network is trained for 5 Monte Carlo simulations using randomly initialized weights.

Post-processing and Segmentation

For this dataset the parameters were optimized empirically on the 15 images used as training images. The PDE ran for 288 iterations with δt equal to 0. 1875, α equal to 0. 1, β equal to 0. 6, and η equal to 1. This places the most weight on the inverse diffusion-based growth term, significant weight on the gradient term, and minimal weight on the curvature term. Following 288 iterations of the pde, the result was thresholded with a threshold of 0. 4 and all of the membrane areas were replaced with their values from a denoised version of the original image. The denoising was done using a nonlocal means denoising algorithm. Following this replacement the watershed process was started using the resultant image. For the watershed merge tree classification, the initial water level was 5 % of the maximum value in each probability map. Due to large section size, we merged regions smaller than $n_r = 300$ pixels with their neighbors in the preprocessing step; 7×7 texture patches were extracted from the original EM images for generating the texton dictionary and building texton histograms as boundary classifier features. A random forest with 500 trees was trained for boundary classification. Figure 10.14 shows the results of the ANN series and post-processing methods for four different test images. Table 10.1 shows the pixel and Rand error of the series ANN model alone and ANN series model followed by PDE post-processing and watershed merge tree segmentation. Notice that while the post-processing worsens the pixel accuracy slightly, it significantly improves the Rand error. Since Rand error is a measure of segmentation errors, this particular trade-off between the pixel accuracy and Rand error is desirable.

Drosophila Ventral Nerve Cord ssTEM

Dataset

The second dataset we experimented with is a stack of 60 images from an ssTEM dataset of the Drosophila first instar larva VNC [8, 108]. It has a resolution of $4 \times 4 \times 50$ nm/pixel and each 2D section is 512×512 pixels. The corresponding binary labels were annotated by an expert neuroanatomist. During the International Symposium on Biomedical Imaging (ISBI) Electron Microscopy Image Segmentation Challenge 30 images were used for training and the remaining images were used for testing.

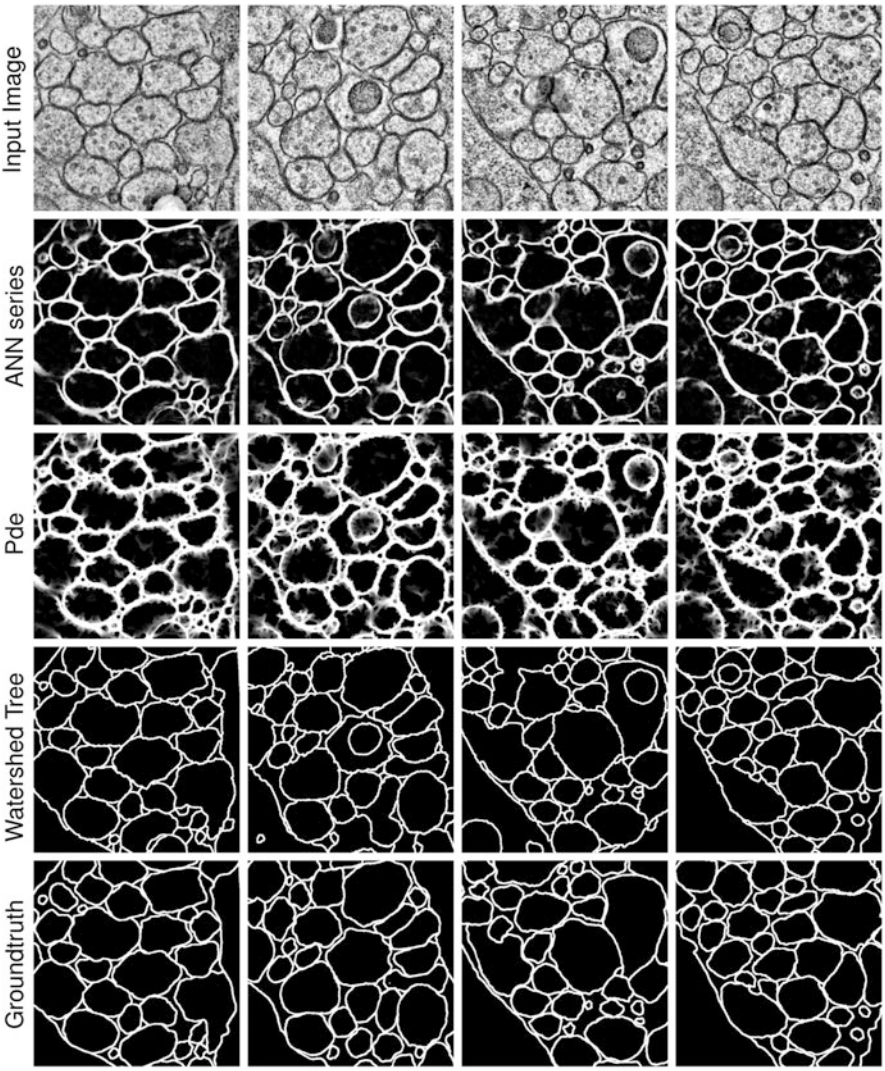

Fig. 10.14 Test results for membrane detection for four different input images from *C. elegans* VNC. The first row shows the input images, row 2 shows the conventional ANN series results, row 3 shows the PDE post-processing results (applied on the results in row 2), and row 4 shows the watershed merge tree results (applied on the results in row 3), and the last row shows the corresponding groundtruth images. All the images in this figure are zoomed in for the sake of better visualization of the details

Table 10.1 Testing performance of the ANN series model and post-processing methods (pde + watershed merge tree) for the *C. elegans* ssTEM dataset

	Training		Testing	
Method	Rand error	Pixel error	Rand error	Pixel error
Series ANN	0.2113	0.0327	0.2285	0.0324
Post-processing	0.0986	0.0431	0.1498	0.0432

Multi-scale Series-ANN Pixel Classifier

In this experiment, a series classifier with five stages was trained using multi-scale contextual method. Each MLP-ANN in the series had one hidden layer with ten nodes. To optimize the network performance, five million samples were randomly selected from the training images such that the training set contained twice the number of negative samples, i.e., the non-membrane samples, than positive samples, i.e., membrane samples. To compute the feature vector for the input image pixels, an 11 by 11 stencil was used to sample the input image and the RLF maps for cell boundaries (at two scales) and mitochondria. The first classifier was trained using this feature vector of size 164. The context features were computed using 5 by 5 patches at four scales (one at original resolution and three at coarser scales) that made the context feature vector of length 100. The remaining classifiers in the series had feature vector of size 264, which included both the input image features and the context features. The evolution of the results through the stages of multi-scale contextual model is shown in Fig. 10.16. It can be seen that the classifier is able to remove some undesired parts such as mitochondria from the interior of the cells.

Post-processing and Segmentation

For this dataset the number of iterations for the PDE post-processing was again optimized empirically using the 30 training images. The optimal number of iterations for this dataset was found to be 425. The remaining parameters were found to be the same because of the similarity in structure between the datasets. Following 425 iterations of the PDE, the result was thresholded with a threshold of 0 and all of the membrane areas were replaced with their values from a nonlocal means [101] denoised version of the original image. Following this replacement the watershed process was started using the resultant image. As for the watershed merge tree method, the initial water level was set as 1 % of the maximum value in each corresponding probability map. Regions smaller than $n_r = 50$ pixels were removed in the initial segmentation; 7×7 texture patches were used for generating texton features. A random forest with 500 trees was again used.

Table 10.2 illustrates the pixel accuracy and Rand error of the multi-scale contextual model alone and multi-scale contextual model followed by PDE

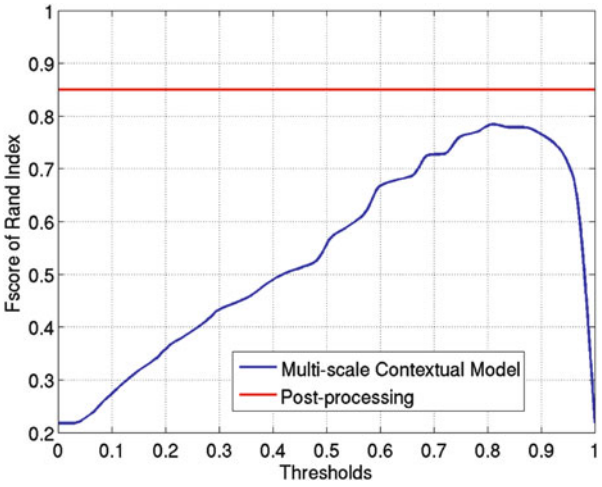

Fig. 10.15 F-value of Rand Index for the testing set in *C. elegans* VNC dataset

post-processing and watershed merge tree segmentation. Similar to section "*C. elegans* Ventral Nerve Cord ssTEM", post-processing worsens the pixel accuracy, it significantly improves the Rand error. Figure 10.17 illustrates the results of the various steps visually.

Figure 10.15 shows the improvement in Rand error provided by the post-processing step against the Rand error of the multi-scale contextual model at different thresholds.

Mouse Neuropil SBFSEM

Dataset

This dataset is a stack of 400 images from the mouse neuropil acquired using SBFSEM. It has a pixel resolution of 10 $\times$10 $\times$10 nm and each 2D section is 4, 096 by 4, 096 pixels. To train and test the segmentation framework, a subset of this data (700 $\times$700 $\times$70) was manually annotated by an expert electron microscopist. From those 70 images, 14 images were used for training and the 56 remaining images were used for testing. The training set contains 4. 5 million samples, which one third of them are positive samples and the remaining of them are negative samples.

Multi-scale Series-ANN Pixel Classifier

The same series classifier as the previous section was trained on this dataset. Figure 10.18 shows four examples of test images and their corresponding

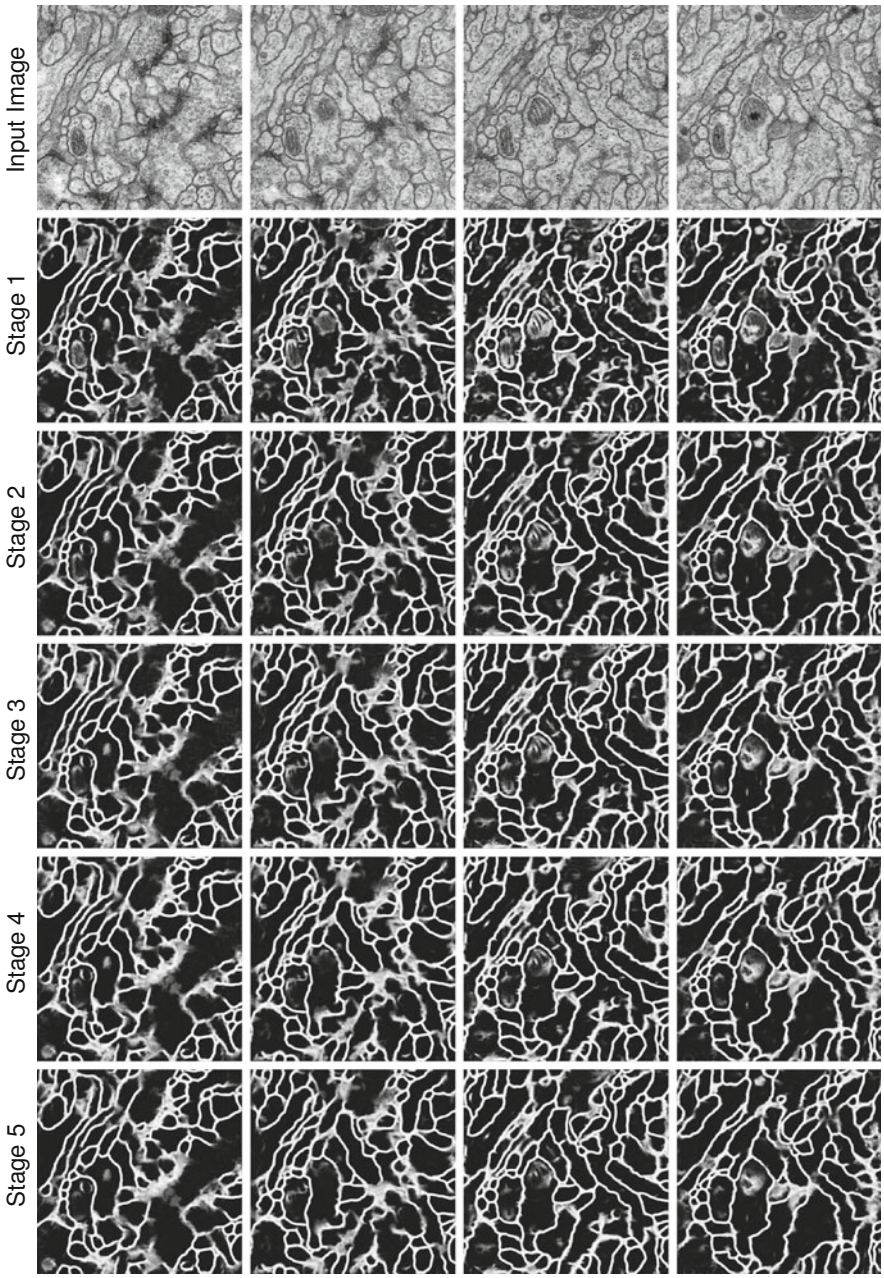

Fig. 10.16 Test results for membrane detection for four different input images from Drosophila VNC. The first row shows the input images, rows 2–5 show the series output at different stages of the multi-scale contextual model

Table 10.2 Testing performance of the multi-scale contextual model and post-processing methods (pde + watershed merge tree) for the Drosophila VNC ssTEM dataset

	Training		Testing	
Method	Rand error	Pixel error	Rand error	Pixel error
Multi-scale contextual model	0.2084	0.0527	0.1312	0.0752
Post-processing	0.0378	0.0599	0.0770	0.1026

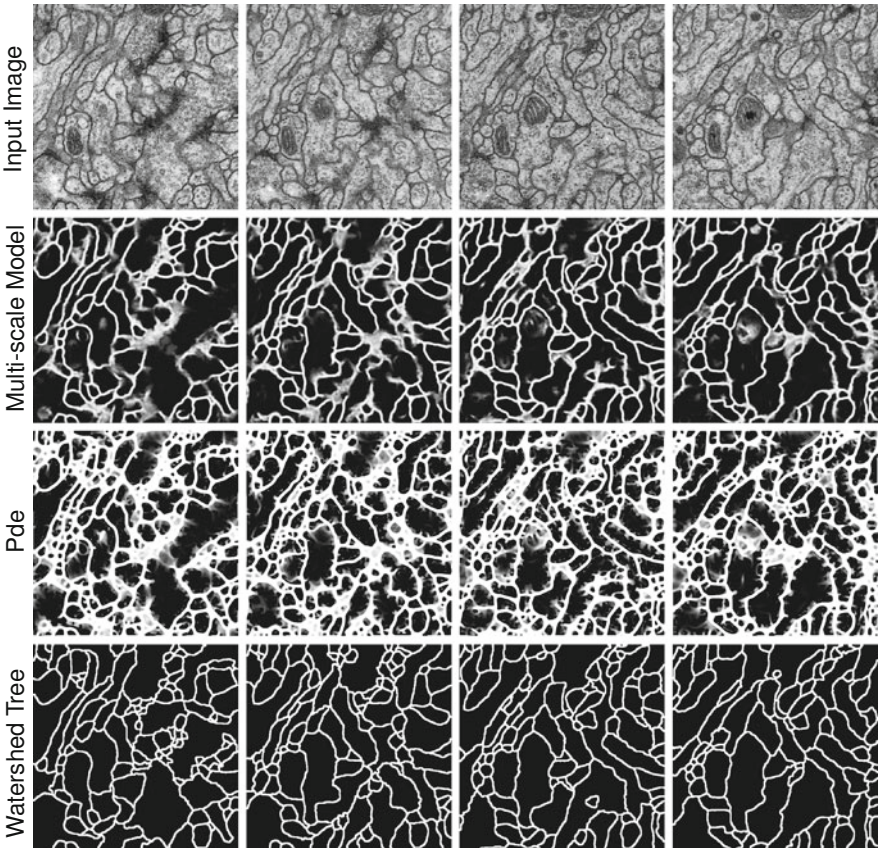

Fig. 10.17 Test results for membrane detection for four different input images from Drosophila VNC. The first row shows the input images, row 2 shows the multi-scale contextual model results, row 3 shows the PDE post-processing results (applied on the results in row 2), and row 4 shows the watershed merge tree results (applied on the results in row 3). The testing ground truth images for the ISBI challenge were not distributed to the contestants; therefore, we are unable to show them here

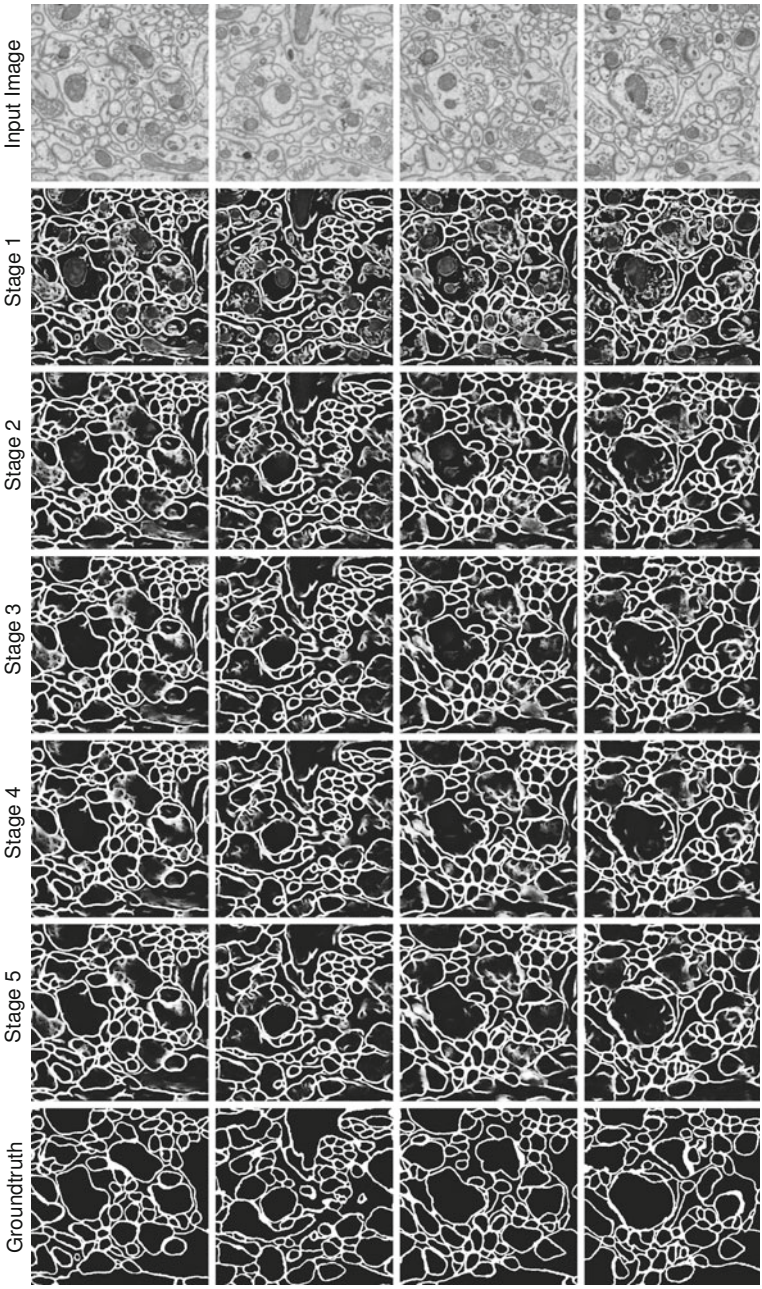

Fig. 10.18 Test results for membrane detection for four different input images from mouse neuropil. The first row shows the input images, rows 2–6 show the series output at different stages, and the last row shows the manually marked image

segmentation results at different stages of the multi-scale contextual model. The accuracy of the results is improved through the stages and cleaner images are obtained at later stages of the series (see Fig. 10.18). Four test images and corresponding membrane detection results for conventional ANN series and multi-scale contextual model are shown in Fig. 10.19. In comparison, multi-scale contextual model is more successful in removing undesired parts and generating cleaner results.

Post-processing and Segmentation

For this dataset the number of iterations was again optimized empirically using the 14 images from bin 1. The optimal number of iterations for this dataset was again found to be 288 while the remaining parameters remained the same. For this dataset, histogram equalization was performed prior to using the image for gradient calculation. The threshold used prior to replacement with the original image intensities in the membrane areas was again 0. Following this replacement the watershedding process was started using the resultant image. The parameters selection of the watershed merge tree classifier was identical to that of the drosophila VNC ssTEM dataset: the initial water level was 1 % of the maximum values correspondingly; regions smaller than $n_r = 50$ pixels were removed; 7×7 texton patches were used; and the random forest utilized 500 trees.

Figure 10.19 illustrates the results of the various steps visually. Table 10.3 shows the pixel accuracy and Rand error of the multi-scale contextual model alone and multi-scale contextual model followed by PDE post-processing and watershed merge tree segmentation. Again, notice that while the post-processing worsens the pixel accuracy slightly, it significantly improves the Rand error. Since Rand error is a measure of segmentation errors, this particular trade-off between the pixel accuracy and Rand error is desirable. Finally, Fig. 10.20 shows the Rand error of the multi-scale contextual model as a function of the final threshold applied to the classifier output. It also shows the Rand error of the post-processing step.

Discussion

In this chapter, we demonstrated a pipeline that utilizes several machine learning strategies to provide a reasonable solution to the problem of segmenting neurons in electron microscopy images. Both supervised and unsupervised techniques were used. The first step of the pipeline is a pixel-level classifier that attempts to mark each pixel in an image either as membrane or non-membrane. This supervised learning approach has clear advantages over traditional image processing methods that involve no learning: the learning approach can adapt to the data, requires no hand designed features, and outperforms the traditional methods. In our pixel-level classifier, we employed a series of ANNs. The ANNs directly use pixel intensities

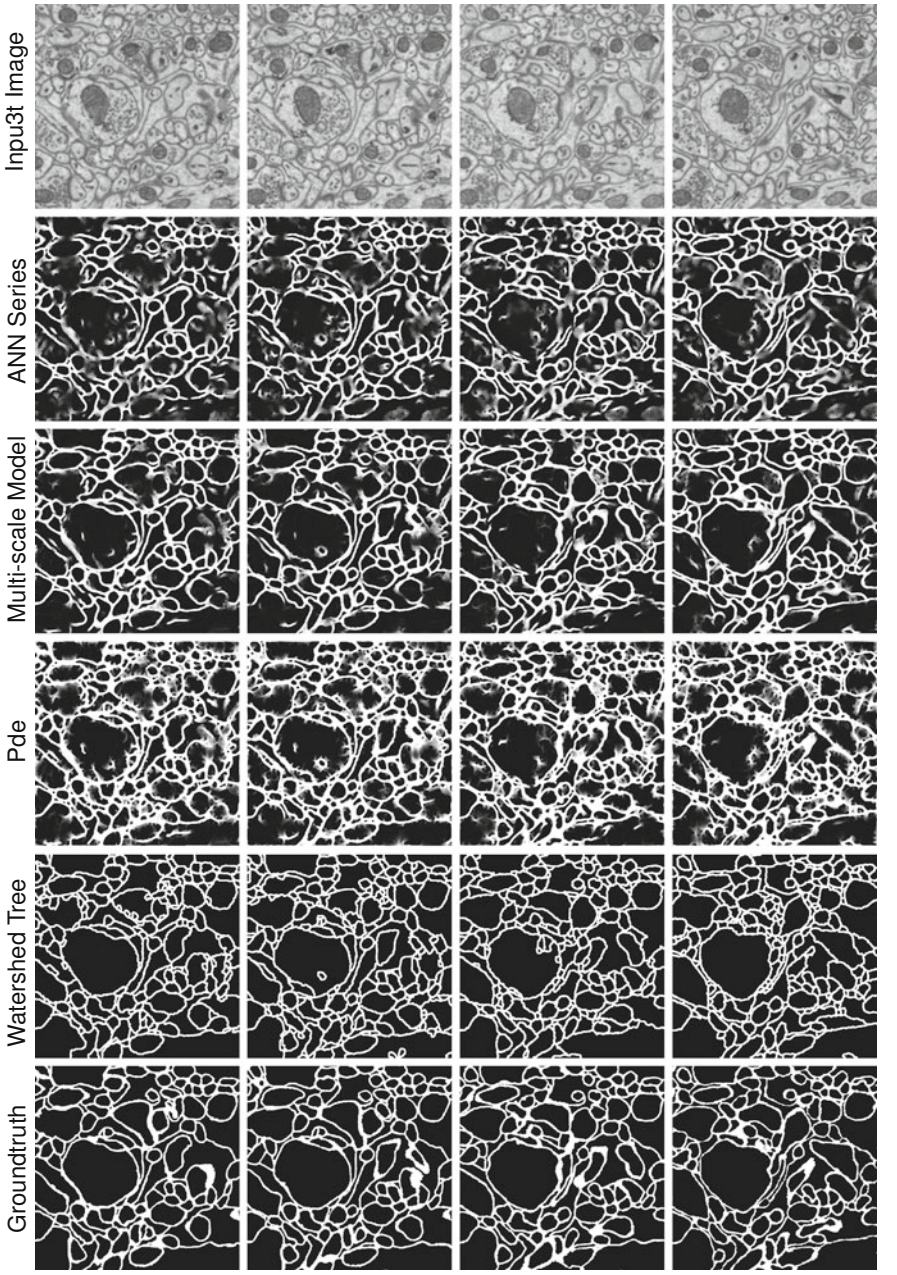

Fig. 10.19 Test results for membrane detection for four different input images from mouse neuropil. The first row shows the input images, the second row shows the conventional ANN series results, row 3 shows the multi-scale contextual model results, row 4 shows the PDE post-processing results (applied on the results in row 3), row 5 shows the watershed merge tree results (applied on the results in row 4), and the last row shows the corresponding grountruth images

Table 10.3 Performance of the multi-scale contextual model and post-processing methods (pde + watershed merge tree) for the mouse neuropil SBFSEM dataset

	Training		Testing	
Method	Rand error	Pixel error	Rand error	Pixel error
Multi-scale contextual model	0.2551	0.0512	0.2413	0.0510
Post-processing	0.1274	0.0716	0.1538	0.0745

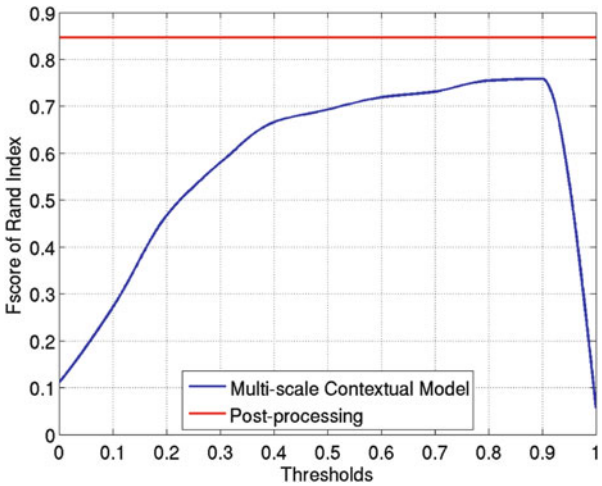

Fig. 10.20 F-value of Rand Index for different thresholds for the mouse neuropil SBFSEM dataset

in neighborhoods as features. Each ANN in the series learns to improve upon the previous ANNs's results by sampling neighborhoods from both the input image and the previous ANN's output (probability map). Intracellular structures such as mitochondria and synaptic vesicles are mostly removed from the membrane detection results and some small gaps in the membranes are filled. These improvements are observed both in training and in testing datasets. A multi-scale version of the series-ANN which can more effectively sample the images was shown to provide further improvement in accuracy. However, the main problem with the pixel-level classifiers is two-fold: (i) it optimizes pixel accuracy instead of segmentation accuracy and (ii) it cannot use region-based features. The first point is problematic because even a single pixel gap in the membrane map, which is negligible in view of pixel error, can create a large segmentation error. The next two steps of our pipeline aim to fix these problems. The second step is an unsupervised PDE which aims to fill small gaps in the membrane map hence favoring over-segmentation over under-segmentation. This is achieved mainly by using an inverse diffusion term

directed along the eigenvector of the Hessian of the probability map associated with its larger eigenvalue, i.e. the direction perpendicular to the cell membrane. The over-segmentation created by the PDE step is finally fixed using the watershed merge tree. Similar to the pixel classifier step, this is also a supervised learning-based step. Incrementally raising the water level in the watershed algorithm creates a hierarchy of regions. Each time two regions merge in the tree due to rising water level, we have to ask the question: Is this merge salient or not? We train a classifier that we call the boundary classifier to answer this question. The advantage of the boundary classifier over the pixel-level classifier is that it can make use of potentially more powerful region-based features. Furthermore, the boundary classifier is trained to optimize segmentation accuracy as measure by the rand error. The saliency of each region is then defined as the probability that its children merge times 1 minus the probability that it doesn't merge with its sibling. Finally, we pick the salient regions from the watershed tree by using a greedy approach in search of the most salient regions. As expected the outcome of these post-processing steps significantly improves segmentation error over the pixel-level classifier. The pixel accuracy is slightly worsened over the pixel-level classifier; however, this is not considered important since the main goal is to improve better segmentations. One reason for the worsening of pixel accuracy is errors in the membrane width that might be introduced by the post-processing which optimizes segmentation error since such errors don't impact the segmentation error.

The pipeline introduced in this chapter can be used to automatically segment neurons in electron microscopy images. However, if perfect accuracy is required as is the case for most connectomics problems, manual proof-reading of the results by an expert will be necessary. To minimize the time that experts need to spend proof-reading results, future methods will focus on further improving the accuracy. Our pipeline also is a new example for how machine learning methods can benefit automatic image analysis.

Acknowledgments This work was supported by NIH R01 EB005832 and 1R01NS075314. The *C. elegans* dataset was provided by the Jorgensen Lab at the University of Utah. The mouse neuropil dataset was provided by the National Center for Microscopy Imaging Research. The retina dataset was provided by the Marc Lab at the University of Utah. The drosophila VNC dataset was provided by the Cardona Lab at HHMI Janelia Farm.

References

1. Suzuki K, Horiba I, Sugie N (2003) Neural edge enhancer for supervised edge enhancement from noisy images. IEEE Trans Pattern Anal Mach Intell 25:1582–1596
2. Suzuki K, Horiba I, Sugie N, Nanki M (2004) Extraction of left ventricular contours from left ventriculograms by means of a neural edge detector. IEEE Trans Med Imag 23:330–339
3. Sporns O, Tononi G, Ktter R (2005) The human connectome: A structural description of the human brain. PLoS Comput Biol 1:e42
4. Briggman KL, Denk W (2006) Towards neural circuit reconstruction with volume electron microscopy techniques. Curr Opin Neurobiol 16:562–570

5. Mishchenko Y (2008) Automation of 3D reconstruction of neural tissue from large volume of conventional serial section transmission electron micrographs. J Neurosci Methods 176 (2):276–289
6. Anderson J, Jones B, Yang J-H, Shaw M, Watt C, Koshevoy P, Spaltenstein J, Jurrus E, U V K, Whitaker R, Mastronarde D, Tasdizen T, Marc R (2009) A computational framework for ultrastructural mapping of neural circuitry. PLoS Biol 7(3):e74
7. Mishchenko Y, Hu T, Spacek J, Mendenhall J, Harris KM, Chklovskii DB (2010) Ultrastructural analysis of hippocampal neuropil from the connectomics perspective. Neuron 67:1009–1020
8. Cardona A, Saalfeld S, Preibisch S, Schmid B, Cheng A, Pulokas J, Tomančák P, Hartenstein V (2010) An integrated micro- and macroarchitectural analysis of the *Drosophila* brain by computer-assisted serial section electron microscopy. PLoS Biol 8(10):e1000502
9. Bock DD, Lee W-C, Kerlin AM, Andermann ML, Hood G, Wetzel AW, Yurgenson S, Soucy ER, Kim HS, Reid RC (2011) Network anatomy and in vivo physiology of visual cortial neurons. Nature 471:177–182
10. Briggman KL, Helmstaedter M, Denk W (2011) Wiring specificity in the direction-selectivity circuit of the retina. Nature 471:183–188
11. Marc RE, Jones BW, Watt CB, Vazquez-Chona F, Vaughan DK, Organisciak DT (2008) Extreme retinal remodeling triggered by light damage: implications for age related macular degeneration. Mol Vis 14:782–806
12. Marc RE, Jones BW, Watt CB, Strettoi E (2003) Neural remodeling in retinal degeneration. Progr Retin Eye Res 22:607–655
13. Sutula T (2002) Seizure-induced axonal sprouting: assessing connections between injury, local circuits, and epileptogenesis. Epilepsy Current 2:86–91
14. Koyama R, Yamada MK, Fujisawa S, Katoh-Semba R, Matsuki N, Ikegaya Y (2004) Brain-derived neurotrophic factor induces hyperexcitable reentrant circuits in the dentate gyrus. J Neurosci 24:7215–7224
15. Xiao YP, Wang Y, Felleman DJ (2003) A spatially organized representation of colour in macaque cortical area v2. Nature 421(6922):535–539
16. Minsky M (1961) Microscopy apparatus. U.S. Patent number 301,467, December 1961
17. Denk W, Strickler JH, Webb WW (1990) Two-photon laser scanning microscopy. Science 248:73–76
18. Egner A, Hell SW (2005) Fluorescence microscopy with super-resolved optical sections. Trends Cell Biol 15:207–215
19. Rust MJ, Bates M, Zhuang X (2006) Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm). Nat Meth 3:793–796
20. Betzig E, Patterson G, Sougrat R, Lindwasser O, Olenych S, Bonifacino J, Davidson M, Lippincott-Schwartz J, Hess H (2006) Imaging intracellular fluorescent proteins at nanometer resolution. Science 313(5793):1642–1645
21. White JG, Southgate E, Thomson JN, Brenner S (1986) The structure of the nervous system of the nematode caenorhabditis elegans. Philos Trans R Soc Lond B Biol Sci 314 (1165):1–340
22. Hall DH, Russell RL (1991) The posterior nervous system of the nematode caenorhaditis elegans: Serial reconstruction of identified neurons and complete pattern of synaptic interactions. J Neurosci 11(1):1–22
23. Chen BL, Hall DH, Chklovskii DB (2006) Wiring optimization can relate neuronal structure and function. Proc Natl Acad Sci USA 103(12):4723–4728
24. Chklovskii DB, Vitaladevuni S, Scheffer LK (2010) Semi-automated reconstruction of neural circuits using electron microscopy. Curr Opin Neurobiol 20(5):667–675
25. Anderson JR, Jones BW, Watt CB, Shaw MV, Yang JH, Demill D, Lauritzen JS, Lin Y, Rapp KD, Mastronarde D, Koshevoy P, Grimm B, Tasdizen T, Whitaker R, Marc RE (2011) Exploring the retinal connectome. Mol Vis 17:355–379

26. Varshney LR, Chen BL, Paniagua E, Hall DH, Chklovskii DB (2011) Structural properties of the *Caenorhabditis elegans* neuronal network. PLoS Comput Biol 7(2):e1001066
27. Deerinck TJ, Bushong EA, Thor A, Ellisman MH (2010) NCMIR methods for 3D EM: A new protocol for preparation of biological specimens for serial block face scanning electron microscopy. Microscopy and Microanalysis Meeting, Portland, OR, 1–5 August 2010
28. Hayworth K, Kasthuri N, Schalek R, Lichtman J (2006) Automating the collection of ultrathin serial sections for large volume TEM reconstructions. Microsc Microanal 12 (2):86–87
29. Tasdizen T, Koshevoy P, Grimm BC, Anderson JR, Jones BW, Watt CB, Whitaker RT, Marc RE (2010) Automatic mosaicking and volume assembly for high-throughput serial-section transmission electron microscopy. J Neurosci Meth 193(1):132–144
30. Saalfeld S, Cardona A, Hartenstein V, Tomančák P (2010) As-rigid-as-possible mosaicking and serial section registration of large ssTEM datasets. Bioinformatics 26(12):i57–i63
31. Tasdizen T, Koshevoy P, Grimm B, Anderson J, Jones B, Watt C, Whitaker R, Marc R (2010) Automatic mosaicking and volume assembly for high-throughput serial-section transmission electron microscopy. J Neurosci Meth 193:132–144
32. Anderson J, Mohammed S, Grimm B, Jones B, Koshevoy P, Tasdizen T, Whitaker R, Marc R (2011) The viking viewer for connectomics: scalable multi-user annotation and summarization of large volume data sets. J Microscopy 241:13–28
33. Knott G, Marchman H, Wall D, Lich B (2008) Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. J Neurosci 28(12):2959–2964
34. Soto GE, Young SJ, Martone ME, Deerinck TJ, Lamont S, Carragher BO, Hama K, Ellisman MH (1994) Serial section electron tomography: A method for three-dimensional reconstruction of large structures. NeuroImage 1(3):230–243
35. Chen X, Winters CA, Reese TS (2008) Life inside a thin section: Tomography. J Neurosci 28 (38):9321–9327
36. Hama K, Arii T, Katayama E, Marton M, Ellisman MH (2004) Tri-dimensional morphometric analysis of astrocytic processes with high voltage electron microscopy of thick golgi preparations. J Neurocytol 33:277–285. doi:10.1023/B:NEUR.0000044189.08240.a2
37. Kreshuk A, Straehle CN, Sommer C, Koethe U, Cantoni M, Knott G, Hamprecht FA (2011) Automated detection and segmentation of synaptic contacts in nearly isotropic serial electron microscopy images. PLoS One 6(10):e24899
38. Andres B, Köthe U, Helmstaedter M, Denk W, Hamprecht FA (2008) Segmentation of SBFSEM volume data of neural tissue by hierarchical classification. In: Rigoll G (ed) Pattern Recognition. LNCS, vol 5096. Springer, Berlin, Heidelberg, pp 142–152
39. Jain V, Murray J, Roth F, Turaga S, Zhigulin V, Briggman K, Helmstaedter M, Denk W, Seung H (2007) Supervised learning of image restoration with convolutional networks. IEEE 11th International Conference on Computer Vision, pp 1–8, Rio de Janeiro, Brazil, 14–21 October 2007
40. Jeong W-K, Beyer J, Hadwiger M, Blue R, Law C, Vazquez-Reina A, Reid RC, Lichtman J, Pfister H (2010) Secret and neurotrace: Interactive visualization and analysis tools for large-scale neuroscience data sets. IEEE Comput Graph 30(3):58–70
41. Jurrus E, Whitaker R, Jones B, Marc R, Tasdizen T (2008) An optimal-path approach for neural circuit reconstruction. In: Proceedings of the 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro, pp 1609–1612, Paris, France, 14–17 May 2008
42. Macke J, Maack N, Gupta R, Denk W, Schölkopf B, Borst A (2008) Contour-propagation algorithms for semi-automated reconstruction of neural processes. J Neurosci Meth 167:349–357
43. Allen BA, Levinthal C (1990) Cartos II semi-automated nerve tracing: Three-dimensional reconstruction from serial section micrographs. Comput Med Imag Graph 14(5):319–329
44. Jurrus E, Tasdizen T, Watanabe S, Davis MW, Jorgensen EM, Whitaker RT (2008) Semi-automated reconstruction of the neuromuscular junctions in the c. elegans. In: MICCAI

Workshop on Microscopic Image Analysis with Applications in Biology, New York, NY, 5–6 September 2008
45. Jurrus E, Watanabe S, Paiva A, Ellisman M, Jorgensen E, Tasdizen T (2012) Semi-automated neuron boundary detection and slice traversal algorithm for segmentation of neurons from electron microscopy images. Neuroinformatics 11(1):5-29
46. Funke J, Andres B, Hamprecht FA, Cardona A, Cook M (2011) Multi-hypothesis crf-segmentation of neural tissue in anisotropic em volumes. CoRR abs/1109.2449
47. Anderson JR, Mohamed S, Grimm B, Jones BW, Koshevoy P, Tasdizen T, Whitaker R, Marc RE (2010) The viking viewer for connectomics: scalable multi-user annotation and summarization of large volume data sets. J Microsc 241(1):1328
48. Vazquez L, Sapiro G, Randall G (1998) Segmenting neurons in electronic microscopy via geometric tracing. In: Proceedings of International Conference on Image Processing, pp 814–818, San Diego, CA, 12–15 October 1998
49. Bertalmío M, Sapiro G, Randall G (2000) Morphing active contours. IEEE Trans Pattern Anal Mach Intell 22:733–737
50. Jurrus E, Hardy M, Tasdizen T, Fletcher P, Koshevoy P, Chien CB, Denk W, Whitaker R (2009) Axon tracking in serial block-face scanning electron microscopy. Med Image Anal 13:180–188
51. Reina AV, Miller E, Pfister H (2009) Multiphase geometric couplings for the segmentation of neural processes. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp 2020–2027, Miami, FL, 20–25 June 2009.
52. Jeong W-K, Beyer J, Hadwiger M, Vazquez A, Pfister H, Whitaker RT (2009) Scalable and interactive segmentation and visualization of neural processes in em datasets. IEEE Trans Visual Comput Graph 15(6):1505–1514
53. Vazquez-Reina A, Miller E, Pfister H (2009) Multiphase geometric couplings for the segmentation of neural processes. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp 2020–2027, Miami, FL, 20–25 June 2009
54. Tasdizen T, Whitaker RT, Marc RE, Jones BW (2005) Enhancement of cell boundaries in transmission electron microscopy images. In: Proceedings of International Conference on Image Processing, vol 2, pp 129–132, Genoa, Italy, 11–14 September 2005
55. Kumar R, Va andzquez Reina A, Pfister H (2010) Radon-like features and their application to connectomics. In: IEEE Computer Society Conference on CVPRW, pp 186–193, San Francisco, CA, 13–18 June 2010
56. Akselrod-Ballin A, Bock D, Reid RC, Warfield SK (2009) Improved registration for large electron microscopy images. In: Proceedings of IEEE International Symposium on Biomedical Imaging, pp 434–437, Boston, MA, 28 June–1 July 2009
57. Preibisch S, Saafeld S, Tomancak P (2009) Globally optimal stitching of tiled 3d microscopic image acquisitions. Bioinformatics 25(11):1463–1465
58. Vu N, Manjunath B (2008) Graph cut segmentation of neuronal structures from transmission electron micrographs. In: Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on, pp 725–728, San Diego, CA, 12–15 October 2008
59. Yang H-F, Choe Y (2009) Cell tracking and segmentation in electron microscopy images using graph cuts. In: Proceedings of IEEE International Symposium on Biomedical Imaging: From Nano to Macro, Boston, MA, 28 June–1 July 2009
60. Yang H-F, Choe Y (2009) 3D volume extraction of densely packed cells in em data stack by forward and backward graph cuts. Computational Intelligence for Multimedia Signal and Vision Processing, pp 47–52, Nashville, Tennessee, 30 March–2 April 2009
61. Kaynig V, Fuchs T, Buhmann JM (2010) Neuron geometry extraction by perceptual grouping in ssTEM images. In: IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, 13–18 June 2010
62. Venkataraju KU, Paiva A, Jurrus E, Tasdizen T (2009) Automatic markup of neural cell membranes using boosted decision stumps. In: EEE International Symposium on Biomedical Imaging (ISBI): From Nano to Macro, Boston, MA, 28 June–1 July 2009

63. Jurrus E, Paiva ARC, Watanabe S, Anderson J, Whitaker BWJRT, Jorgensen EM, Marc R, Tasdizen T (2010) Detection of neuron membranes in electron microscopy images using a serial neural network architecture. Med Image Anal 14(6):770–783
64. Geman S, Geman D (1984) Stochastic relaxation, gibbs distributions and the bayesian restoration of images. IEEE Trans Pattern Anal Mach Intell 6:721–741
65. Freeman WT, Pasztor EC, Owen T, Carmichael Y (2000) Merl a mitsubishi electric research laboratory. Int J Comput Vis 40:2000
66. Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: International Conference on Machine Learning, pp 282–289, Williams College, Williamstown, MA, 28 June–1 July 2001
67. Kumar S, Hebert M (2003) Discriminative random fields: A discriminative framework for contextual interaction in classification. In: ICCV, pp 1150–1157, Nice, France, 14–17 October 2003
68. Jain V (2010) Machine Larning of Image Analysis with Convolutional Networks and Topological Constraints. PhD Thesis, MIT
69. Turaga SC, Briggman KL, Helmstaedter M, Denk W, Seung HS (2009) Maximin learning of image segmentation. In: Advances in Neural Information Processing Systems, Vancouver, British Columbia, Canada, 7–10 December 2009
70. Turaga SC, Murray JF, Jain V, Roth F, Helmstaedter M, Briggman KL, Denk W, Seung HS (2010) Convolutional networks can learn to generate affinity graphs for image segmentation. Neural Comput 22(2):511–538
71. Jain V, Bollmann B, Richardson M, Berger DR, Helmstaedter MN, Brigmann KL, Bowden JB, Mendenhall JM, Abraham WC, Harris KM, Kasthuri N, Hayworth KJ, Schalek R, Tapia JC, Lichtmann JW, Seung HS (2010) Boundary learning by optimization with topological constraints. In: IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, 13–18 June 2010
72. Veeraraghavan A, Genkin AV, Vitaladevuni S, Scheffer L, Xu S, Hess H, Fetter R, Cantoni M, Knott G, Chklovskii D (2010) Increasing depth resolution of electron microscopy of neural circuits using sparse tomographic reconstruction. In: IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, 13–18 June 2010
73. Fukushima K (1982) Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. Pattern Recogn 15(6):455–469
74. Hubel D, Wiesel T (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol 160:106–154
75. LeCun Y, Bengio Y (1995) Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks. MIT Press, Cambridge, MA, pp 255–258
76. Garcia C, Delakis M (2004) Convolutional face finder: A neurl architecture for fast and robust face detection. IEEE Trans Pattern Anal Mach Intell 26(11):1408–1423
77. Osadchy R, Miller M, LeCun Y (2005) Synergistic face detection and pos estimation with energy-based model. In: Advances in Neural Information Processing Systems, Vancouver, British Columbia, Canada, 5–8 December 2005
78. Lawrence S, Giles CL, Tsoi AC, Back A (1997) Face recognition: A convolutional neural network approach. IEEE Trans Neural Network 8(1):98–113
79. LeCun Y, Huang FJ, Bottou L (2004) Learning methods for generic object recognition with invariance to pose and lighting. In: Proceedings of the 2004 I.E. Computer Society Conference on Computer Vision and Pattern Recognition, vol 2, pp II–97–104, Washington, DC, 27 June–2 July 2004
80. Huang FJ, LeCun Y (2006) Large-scale learning with svm and convolutional nets for generic object categorization. In: IEEE Conference on Computer Vision and Pattern Recognition, New York, NY, 17–22 June 2006
81. Ning F, Delhomme D, Lecun Y, Piano F, Bottou L, Barbano PE (2005) Toward automatic phenotyping of developing embryos from videos. IEEE Trans Image Process 14:1360–1371

82. Seyedhosseini M, Kumar R, Jurrus E, Guily R, Ellisman M, Pfister H, Tasdizen T (2011) Detection of neuron membranes in electron microscopy images using multi-scale context and radon-like features. In: Medical Image Computing and Computer-Assisted Intervention MICCAI 2011. Lecture Notes in Computer Science (LNCS), vol 6891. Springer, Berlin, Heidelberg, pp 670–677
83. Lee H, Grosse R, Ranganath R, Ng AY (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: International Conference on Machine Learning, vol. 382, pp 609–616, Montreal, Quebec, Canada, 2009
84. Norouzi M, Ranjbar M, Mori G (2009) Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In: IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 20–25 June 2009
85. Hinton GE, Osindero S (2006) A fast learning algorithm for deep belief nets. Neural Comput 18:2006
86. Ciresan D, Giusti A, Gambardella L, Schmidhuber J (2012) Neural networks for segmenting neuronal structures in EM stacks. In: ISBI Electron Microscopy Segmentation Challenge, Barcelona, Spain, 2–5 May 2012
87. Tu Z (2008) Auto-context and its application to high-level vision tasks. IEEE Conference on Computer Vision and Pattern Recognition, pp 1–8, Anchorage, Alaska, 24–26 June 2008
88. Tu Z, (2008) Auto-context and its application to high-level vision tasks. In: Proceedings of IEEE Computer Vision and Pattern Recognition, Anchorage, Alaska, 24–26 June 2008
89. Haykin S (1999) Neural networks: A comprehensive foundation, 2nd edn. Prentice-Hall, Upper Saddle River, NJ
90. Principe JC, Euliano NR, Lefebvre WC (2000) Neural and adaptive systems: Fundamentals through simulations. Wiley, New York
91. Pomerleau D (1993) Knowledge-based training of artificial neural networks for autonomous robot driving. In: Connell J, Mahadevan S (eds) Robot Learning. Kluwer Academic, Dordrecht, pp 19–43
92. Wells G, Venaille C, Torras C (1996) Promising research: Vision-based robot positioning using neural networks. Image Vis Comput 14:715–732
93. Cottrell G (1990) Extracting features from faces using compression networks: face, identity, emotion and gender recognition using holons. In: Connection models: Proceedings of the 1990 summer school. Morgan Kaufmann, San Mateo, CA, pp 328–337
94. Rabi G, Lu S (1998) Visual speech recognition by recurrent neural networks. J Electron Imag 7:61–69
95. Venkatataju KU, Paiva A, Jurrus E, Tasdizen T (2009) Automatic markup of neural cell membranes using boosted decision stumps. In: Proceedings of the 6th IEEE International Symposium on Biomedical Imaging, pp 1039–1042, Boston, MA, 28 June–1 July 2009.
96. Leung T, Malik J (2001) Representing and recognizing the visual appearance of materials using three-dimensional textons. Int J Comput Vision 43(1):29–44
97. Schmid C (2001) Constructing models for content-based image retrieval. In: Computer Vision and Pattern Recognition, 2001, CVPR 2001. Proceedings of the 2001 I.E. Computer Society Conference on, vol. 2, pp II–39–II–45, Kauai, HI, 8–14 December 2001
98. Varma M, Zisserman A (2003) Texture classification: are filter banks necessary? In: Computer Vision and Pattern Recognition, 2003. Proceedings of the 2003 I.E. Computer Society Conference on, vol. 2, pp II–691–8, Madison, WI, 16–22 June 2003
99. Awate SP, Tasdizen T, Whitaker RT (2006) Unsupervised Texture Segmentation with Nonparametric Neighborhood Statistics. In: Proceedings of the European Conference on Computer Vision. pp 494–507, Graz, Austria, 7–13 May 2006
100. Awate SP, Whitaker RT (2006) Unsupervised, information-theoretic, adaptive image filtering for image restoration. IEEE Trans Pattern Anal Mach Intell 28(3):364–376
101. Buades A, Coll B, Morel J-M (2005) A non-local algorithm for image denoising. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp 60–65, San Diego, CA, 20–26 June 2005

102. Tasdizen T (2008) Principal components for non-local means image denoising. In: Proceeding of International Conference on Image Processing, San Diego, California, 12–15 October 2008
103. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
104. Jin Y, Hoskins R, Horvitz HR (1994) Control of type-D GABAergic neuron differentiation by *C. elegans* UNC-30 homeodomain protein. Nature 372:780–783
105. White JQ, Nicholas T, Gritton J, Truong L, Davidson ER, Jorgensen EM (2007) The sensory circuitry for sexual attraction in *C. elegans* males. Curr Biol 17:1847–1857
106. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Contr Signals Syst 2:303–314
107. Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Network 4(2):251–257
108. Cardona A, Saalfeld S, Schindelin J, Arganda-Carreras I, Preibisch S, Longair M, Tomancak P, Hartenstein V, Douglas RJ (2012) Trakem2 software for neural circuit reconstruction. PLoS One 7(6):e38011