Supervised Machine Learning for Greedy Agglomeration in Connectomics

Karan Kathpalia

A Thesis
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Master of Science in Engineering

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Hyunjune Sebastian Seung

June 2017

Abstract

This study explores the use of supervised machine learning methods for greedy agglomeration in the application of constructing connectomes or neural wiring diagrams that show how neurons are connected to each other. The current approach to this problem is mean affinity greedy agglomeration, which makes locally optimal merge/not-merge decisions. The rationale behind using supervised learning methods is that they may lead to locally optimal merge/not-merge decisions that are more globally optimal than those made by mean affinity agglomeration. The results of this study are inconclusive on whether supervised machine learning methods are better than mean affinity agglomeration because of evaluation issues. Future work should thus explore better methods of evaluating agglomeration performance. In addition, future work should address the specific weaknesses of the pipeline for producing connectomes: misalignment errors and errors where the dendritic spines do not grow out from the dendritic shafts.

Acknowledgements

The work presented in this thesis was performed in collaboration with Jonathan Zung, Ignacio Tartavull, Kisuk Lee and Sebastian Seung. I would like to acknowledge their contributions as this thesis would not have been possible without their work. Jonathan and Ignacio built the greedy agglomeration framework, formulated the supervised machine learning approach to greedy agglomeration, implemented the random forest machine learning model and designed the majority of the feature set. Kisuk trained the multi-scale convolutional neural network used to predict affinity maps. Sebastian gave constructive advice about the experimental setup and helped with the graph analysis.

Contents

Chapter 1

Introduction

Connectomics is a field of study that seeks to map the connections between neurons in the brain (the connectome) to understand how these connections shape our thoughts, feelings, perceptions and memories. Please see Figure 1.1 for an illustration of a connectome. The human brain contains over 10 billion neurons and 100 trillion synapses and so it is impractical to construct a connectome manually. At present, only the connectome of *C. elegans* has been fully reconstructed in a labour-intensive, manual process because the brains of *C. elegans* contain roughly 300 neurons [23]. It is thus necessary to perform automatic reconstruction of the connections between the neurons in the human brain in order to map the connectome of a human brain in a feasible amount of time.

Neuroscientists are currently trying to construct partial connectomes of the mouse brain (in particular, the retina and cortex) to test the hypothesis that the connections between neurons in the brain shape our thoughts, feelings, perceptions and memories. The standard approach to the problem of automatic reconstruction is to cut a subsection of a mouse brain into tiny slices, capture high-resolution serial section EM (electron microscopy) images of these slices to create a 3D stack or volume of aligned images, use a multi-scale convolutional neural network to predict voxel-wise affinity maps for the volume in the x,y,z directions, run the watershed algorithm on the affinity maps to create oversegmented supervoxels, perform greedy agglomeration of the oversegmented supervoxels to assemble neuron part volumes and correct the errors in the partially assembled connectome via human proof-reading.

The current approach to greedy agglomeration is to first compute the mean affinity of voxels along the shared boundaries of two supervoxels for every pair of adjacent supervoxels and then repeat the following process until there are no more supervoxels to be merged: find the supervoxel pair with the highest mean affinity and if the mean affinity exceeds a threshold, merge the two supervoxels and update the mean affinities of pairs consisting of the surrounding supervoxels and the merged supervoxels. The goal of this study is to investigate whether a supervised machine learning approach (which performs inference using models trained on labelled data) to greedy agglomeration would lead to the construction of more accurate connectomes.

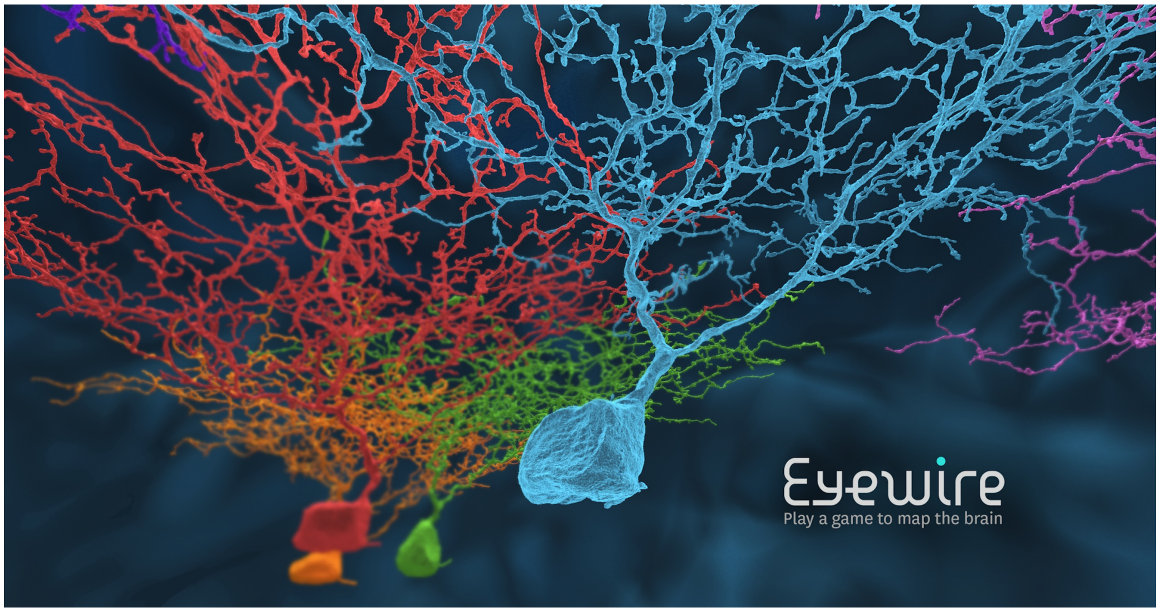

Figure 1.1: Partial connectome of mouse retina constructed in the citizen science game Eyewire [1]

Chapter 2

Related Work

In this paper, convolutional neural networks are used to perform boundary detection and affinity map prediction. Convolutional neural networks are part of a special class of artificial neural networks called multi-layer perceptrons [6]. Artificial neural networks consist of neurons, weighted connections between neurons and a non-linear activation function for the neurons. Multi-layer perceptrons are feedforward neural networks (directed acylic graphs where nodes are neurons and edges are connections) arranged in layers such that each neuron in a given layer is only connected to neurons in the next layer. Typically, each neuron is connected to every neuron in the next layer in multi-layer perceptrons. Artificial neural networks are trained using backpropagation, which computes the partial derivatives of the reward/loss function with respect to each individual weight and bias parameter in the network [19].

Convolutional neural networks were invented at Bell Labs to solve the problem of recognising handwritten digits and letters in images [12]. Convolution is the operation of applying a flipped kernel to a signal by "sliding" it around the image and computing a linear combination of the inputs "covered" by the kernel using the kernel weights. A convolutional layer contains a convolutional kernel for each possible combination of input feature map and output feature map, and applies the appropriate convolutional kernel to the appropriate input feature map to obtain part of the preactivity image for the output feature map. A pooling layer simply down-samples the resolution of the input feature maps, which is typically performed by either taking the maximum or average of input feature map values within some small rectangular grid. A convolutional neural network consists of alternating convolutional layers and pooling layers, followed by a multi-layer perceptron [13].

Deep learning, the training of neural networks with a large number of layers, has become the prevalent approach in computer vision in recent literature. Before 2012, standard supervised machine learning methods and hand-designed features were commonly used in computer vision. This was because the labelled datasets used for training machine learning classifiers for computer vision applications were relatively small before 2012, which prevented high capacity models such as deep neural networks from being properly fitted to the data. The performance capabilities of deep convolutional neural networks in computer vision were finally realised with the introduction of ImageNet, a database of images of objects (i.e. visual categories) containing over 1 million labelled images and over 1000 object classes [5, 20]. In 2012, a deep convolutional neural network model developed by researchers at the University of Toronto won the ImageNet Large-scale Visual Recognition Challenge by a large margin over traditional supervised machine learning models [11]. This development convinced many computer vision researchers to develop large-scale annotated datasets of images and train deep convolutional neural networks on these datasets to solve computer vision problems. At the time of writing, convolutional neural networks have demonstrated high performance at object recognition [11], scene recognition [24], semantic segmentation [15], the playing of classic Atari games [16], and many other applications.

In connectomics, convolutional neural networks are used to perform boundary prediction (i.e. predict which voxels contain a neuronal boundary) and affinity map prediction (i.e. predict affinities or similarities between pairs of adjacent voxels in the x, y, z directions) given EM images of the brain. Using sliding-window deep convolutional neural networks for boundary prediction became prevalent after Ciresan et al. won the ISBI 2012 competition with a deep convolutional neural network that outperformed the other entries by a considerable margin [2, 4]. Convolutional neural networks were first used to predict affinity maps in 2010 before Ciresan et al. won the ISBI 2012 competition and AlexNet won the ImageNet Large-scale Visual Recognition Challenge [22]. Recent literature on deep learning applied to biomedical images (including EM images of the brain) has focused on the use of recursive training of 2D-3D convolutional neural networks (i.e. train a 2D convolutional neural network first, add a 3D convolutional neural network at the end that takes 2D input from the 2D convolutional neural network output along with the raw voxel values, and then train both networks simultaneously) [14], and on the use of deconvolutions (upsampling convolutions) in U-shaped network architectures [18].

The literature in agglomeration for connectomics features a variety of approaches involving reinforcement learning, context-aware delayed agglomeration and unsupervised deep learning. Jain et al. interpret agglomeration as a reinforcement learning problem with value estimation (i.e. use a machine learning method to predict values of the action-value function) and infinite discounting [9]. Their method attempts to learn the similarity function between two superpixels and then use the learned similarity function for agglomeration. On the other hand, Parag et al. use a method that separates superpixels into different semantic classes (cytoplasm and mitochondria) and then performs class-specific agglomeration but some supervoxel merges are delayed until a more confident boundary prediction can be generated [17]. This method uses greedy delayed-agglomeration, and a supervised machine learning method for boundary prediction that helps cluster the cytoplasm superpixels. In contrast to the two aforementioned papers, Bogovic et al. use unsupervised deep learning to learn 3D agglomeration features and show that the automatically learned features combined with hand-designed features lead to better agglomeration performance [3]. Compared to the aforementioned methods, the agglomeration approach taken in this paper is different in the following manner:

- The basic approach is that a multi-scale convolutional neural network is used to predict voxel affinities in the x, y, z directions and the mean affinity of the shared boundary voxels between two adjacent supervoxels is used to decide whether to merge the supervoxels in greedy agglomeration.

- Supervised machine learning is used with some features generated from the affinity map multi-scale convolutional neural network to learn a function that predicts whether or not to merge two adjacent supervoxels.
- A special procedure called teacher-forced mean affinity agglomeration is used to generate labelled examples for the supervised machine learning classifiers using the agglomeration ground truth (i.e. the expected output of agglomeration).

In short, the major contribution of this paper is the use of supervised machine learning methods for merge decisions in greedy agglomeration in conjunction with a novel procedure for generating labelled examples to help train and test these methods.

Chapter 3

Design

This chapter introduces the datasets, methods and experiments used to design the supervised machine learning formulation for greedy agglomeration and evaluate whether this formulation of greedy agglomeration leads to higher quality connectomes.

3.1 Datasets

The SNEMI3D and AC3 datasets are used in experiments in this study. Both datasets are subvolumes of the dataset used in the paper "Saturated Reconstruction of a Volume of Neocortex" [10]. Both datasets contain 3D image stacks of the mouse neocortex taken using serial section Scanning Electron Microscopy (ssSEM). The SNEMI3D dataset contains a training set with 100 EM image sections of size 1024 x 1024 and a test set with 100 EM image sections of size 1024 x 1024. The AC3 dataset contains a training set with 256 EM image sections of size 1024 x 1024 and a test set with 256 EM image sections of size 1024 x 1024. Each dataset contains labels in the following format: each voxel in the dataset is assigned a number that corresponds to some object so that all the voxels inside a particular object have the same number. Please see Figure 3.1, which contains a sample serial section from the SNEMI3D dataset with labels superimposed.

For SNEMI3D, the multi-scale convolutional neural network was trained on the SNEMI3D training set while greedy agglomeration was trained on the training set and tested on the test set. Observe that the convolutional neural network is used to predict affinities on data that it was trained on. To ensure that this does not significantly affect the results, the AC3 test set was split into a training set containing 128 EM image sections and a test set containing 128 EM image sections. The multi-scale convolutional neural network was trained on the original training set while greedy agglomeration was respectively trained and tested on the training and test sets created by partitioning the original test set.

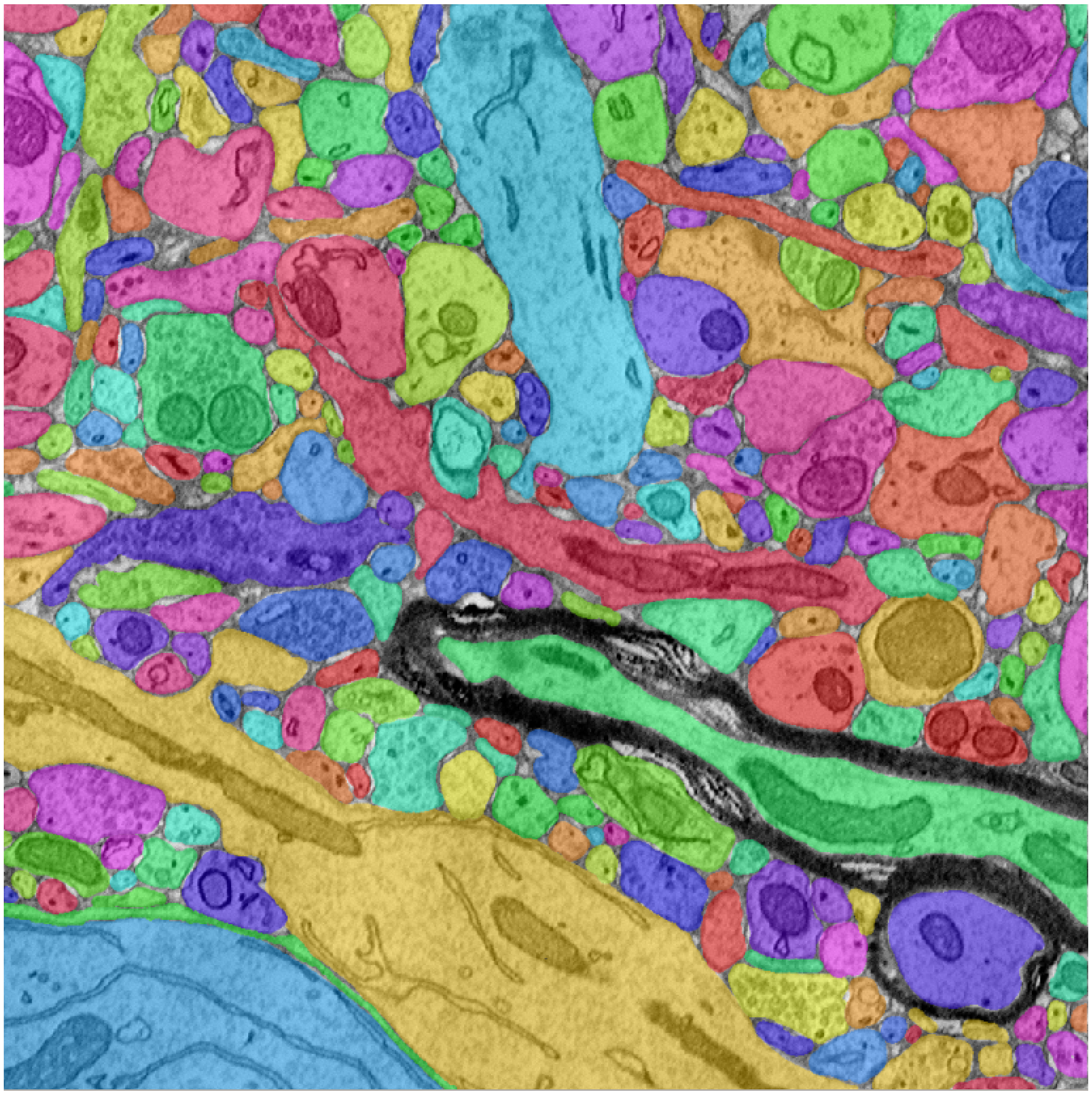

Figure 3.1: Training labels superimposed on serial section image from SNEMI3D dataset [8]

3.2 Methods

3.2.1 Multi-scale Convolutional Neural Network for Boundary Detection and Affinity Map Prediction

A multi-scale convolutional neural network is used to perform boundary detection and affinity map prediction. The two tasks are closely related and similar network architectures are used for both. The difference is that in boundary detection, the network predicts a label from the set {boundary, not-boundary} for each voxel (i.e. detect neuronal boundaries in the stack) [4] while in affinity map prediction, the network predicts affinity maps in the x, y and z directions (the affinity between two voxels indicates how similar the voxels are) [22]. The following description focuses on networks for boundary detection because they are easier to describe. Given a stack of EM images, each training iteration randomly samples a mini-batch of 3D context windows centred at specific voxels in the stack along with the true labels for these specific voxels and performs backpropagation to obtain the weight updates for each neuron in the network. The multi-scale neural network was trained using mini-batch stochastic gradient descent with momentum, dropout regularisation, a manual learning rate schedule (i.e. drop the learning rate by a factor of 2 every time the validation error flatlines until this is no longer effective) and data augmentation (rotation, horizontal flipping and vertical flipping of context windows) [7, 11, 21]. The convolutional neural network makes multi-scale predictions because there are different parallel paths in the network architecture with different receptive fields that are combined at the end to give the final prediction (i.e. branching architecture).

3.2.2 Mean Affinity Agglomeration

The mean affinity agglomeration algorithm is a greedy agglomeration algorithm that starts off by computing the mean affinity of the supervoxels along the shared boundary of every two adjacent supervoxels. The algorithm then repeats the following steps until no more supervoxels can be merged:

1. Find the supervoxel pair with the highest mean affinity. Break ties arbitrarily.

2. If the mean affinity of that pair exceeds a threshold t, merge the two supervoxels in the pair and update the mean affinities of the merged supervoxels and the adjacent supervoxels.

3.2.3 Supervised Greedy Agglomeration

Machine Learning Methodology

Greedy agglomeration is formulated as a machine learning task in the following manner: given two adjacent supervoxels as input, make a merge or not-merge decision and give the degree of confidence in that decision. The confidence is given as a real number in the interval $[0, 1]$ where 0 represents 0% confidence in merging and 1 represents 100% confidence in merging. This task can be interpreted as both a binary classification task (predict 0/1 labels and the confidence) and a regression task (predict the merge confidence directly). The following machine learning methods are used for supervised greedy agglomeration: logistic regression (classification), random forest (regression) and linear regression (regression).

Labelled training and test datasets for greedy agglomeration were generated using teacher-forced mean affinity agglomeration. Teacher-forced mean affinity agglomeration follows the same process as the mean affinity agglomeration algorithm described above but the agglomerator is prevented from making any mistakes (incorrect merge/non-merge decisions). For each supervoxel pair considered for merging, the features and the correct label are recorded. Teacher-forced mean affinity agglomeration is respectively performed on the training and test images to get the supervised greedy agglomeration training and test sets.

Each record or example in the supervised greedy agglomeration datasets comes with a set of features and a label. The following features are computed for each supervoxel pair under consideration: the mean affinity of the voxels along the shared boundary, the maximum affinity of the voxels along the shared boundary, the log10 of the minimum volume of the two supervoxels in the pair, the log10 of the maximum volume of the two supervoxels in the pair and the log10 of the amount of contact area between the two supervoxels in the pair. For logistic regression, the features are standardised to have a mean of 0 and standard deviation of 1 by subtracting the feature means from the original feature vector and dividing the resulting vector by the feature standard deviations. Furthermore, the logistic regression agglomerator is only trained using examples with labels at most 0.1 or at least 0.9.

Recall that the labels are real numbers in the interval $[0, 1]$. For each supervoxel pair under consideration, the label is computed using the steps:

1. Find the volume of the intersection of the first supervoxel with each ground truth segment (each possible label) and compute a vector with length equal to the number of ground truth segments (the number of labels).

2. Find the volume of the intersection of the second supervoxel with each ground truth segment (each possible label) and compute a vector with length equal to the number of ground truth segments (the number of labels).

3. Normalise both vectors to have unit length.

4. Compute the dot product between the two vectors to obtain the label for supervised greedy agglomeration.

The intuition behind this labelling technique is that the dot product is 1 if both supervoxels are completely inside the same ground truth segment. The dot product is 0 if both supervoxels do not intersect with any common ground truth segment. In practice, this labelling technique leads to labels that are close to 0 or 1.

Greedy Agglomeration Algorithm

The supervised greedy agglomeration is initialised by computing the merge confidence of every two adjacent supervoxels. The algorithm then repeats the following steps until no more supervoxels can be merged:

1. Find the supervoxel pair with the highest merge confidence. Break ties arbitrarily.

2. If the merge confidence exceeds a threshold t, merge the two supervoxels in the pair and update the merge confidences of the merged supervoxels and the adjacent supervoxels.

3.3 Experiments

3.3.1 Classification Precision-Recall Graphs

This set of experiments is intended to evaluate the classification performance of the machine learning models (including mean affinity agglomeration) on the supervised greedy agglomeration datasets (both the training and test sets) generated using the SNEMI3D and AC3 datasets. The classification task involves predicting whether or not to merge a given supervoxel pair. Note that the regression methods have their output rounded to the nearest whole number. The machine learning methods are evaluated using graphs with precision-recall curves (one curve per method) where a higher curve means better performance. The vertical axis represents precision while the horizontal axis represents recall. For a given threshold t, the precision and recall of each machine learning method is computed. The precision is the number of true positives over the sum of the number of true positives and the number of false positives (positive = merge, negative = not-merge). The recall is the number of true positives over the sum of the number of true positives and false negatives. For each machine learning method, the precision and recall are computed for thresholds $t \in \{0.05 : 0.05 : 0.95\}$ to obtain the data required to draw a full precision-recall curve.

3.3.2 Agglomeration Precision-Recall Graphs

This set of experiments is intended to evaluate the agglomeration performance of the machine learning models (including mean affinity agglomeration) on the SNEMI3D and AC3 datasets (both the training and test sets). The output of agglomeration is a set of supervoxels, which correspond to neuron parts. The metrics of precision and recall are used to evaluate agglomeration performance. However, precision and recall are defined differently for agglomeration in a manner that is similar to how the Rand index is computed. The Rand index numerator is the sum of the number of voxel pairs in the same set in both the predicted segmentation and the ground truth segmentation, and the number of voxel pairs in different sets in both the predicted segmentation and the ground truth segmentation. The Rand index denominator is the number of combinations of two voxels that can be formed from all the voxels in the volume. The Rand index is thus computed by dividing the Rand index numerator by the Rand index denominator.

The agglomeration precision is computed by dividing the number of voxel pairs in the same segment in both the predicted segmentation and ground truth segmentation by the number of voxel pairs which the predicted segmentation reports are in the same segment. The agglomeration recall is computed by dividing the number of voxel pairs in the same segment in both the predicted segmentation and ground truth segmentation by the number of voxel pairs which the ground truth segmentation reports are in the same segment.

Greedy agglomeration starts with a high threshold and continues with that threshold until no more supervoxels can be merged. When no more supervoxels can be merged, the threshold is decreased. Greedy agglomeration then continues in the same manner and stops completely when the threshold reaches 0. The precision and recall are computed every 100 merges during greedy agglomeration. This process generates enough data to plot a roughly full precision-recall curve for each machine learning method. All the precision-recall curves are plotted on a graph with precision on the vertical axis and recall on the horizontal axis.

Chapter 4

Evaluation

4.1 Classification Precision-Recall Graphs

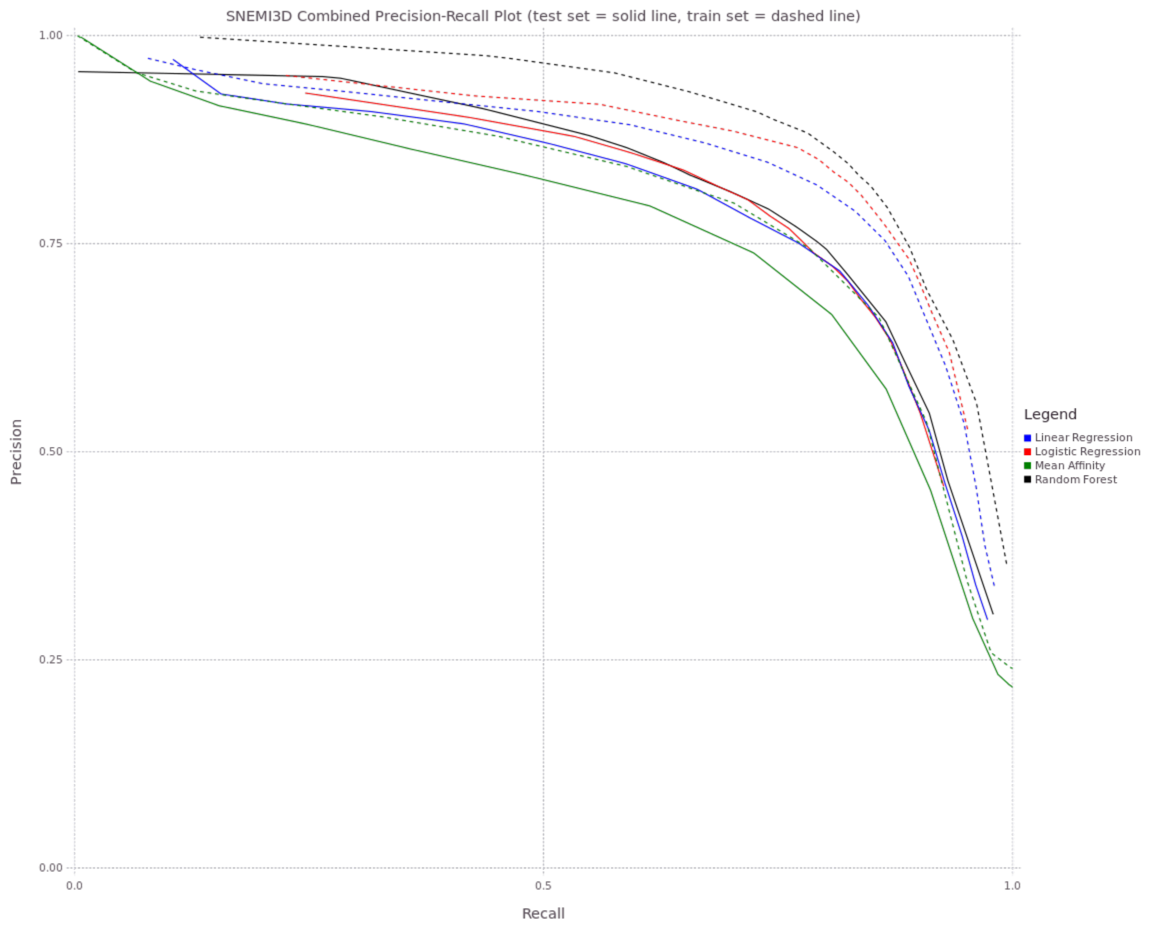

Figure 4.1: Classification precision-recall graph for the SNEMI3D dataset

As seen in Figure 4.1, all the methods have better classification performance on the SNEMI3D training set than on the SNEMI3D test set (higher curve means better performance). Out of all the methods, random forest has the best classification performance on both the training and test set. However, random forest only has significantly better classification performance than the other supervised machine learning models on the training set and slightly better classification performance than the other supervised machine learning models on the test set. On the training set, all the supervised machine learning methods give better classification performance than the mean affinity agglomeration baseline. On the test set, random forest, linear regression and logistic regression have similar classification performance and they all have significantly higher classification performance than the mean affinity agglomeration baseline.

On the other hand, Figure 4.2 indicates that all the methods have comparable classification performance on the AC3 training and test sets. It appears that random forest is slightly better than the other methods but this is difficult to ascertain. The machine learning methods perform significantly better than the mean affinity agglomeration baseline on the SNEMI3D test set but not on the AC3 test set.

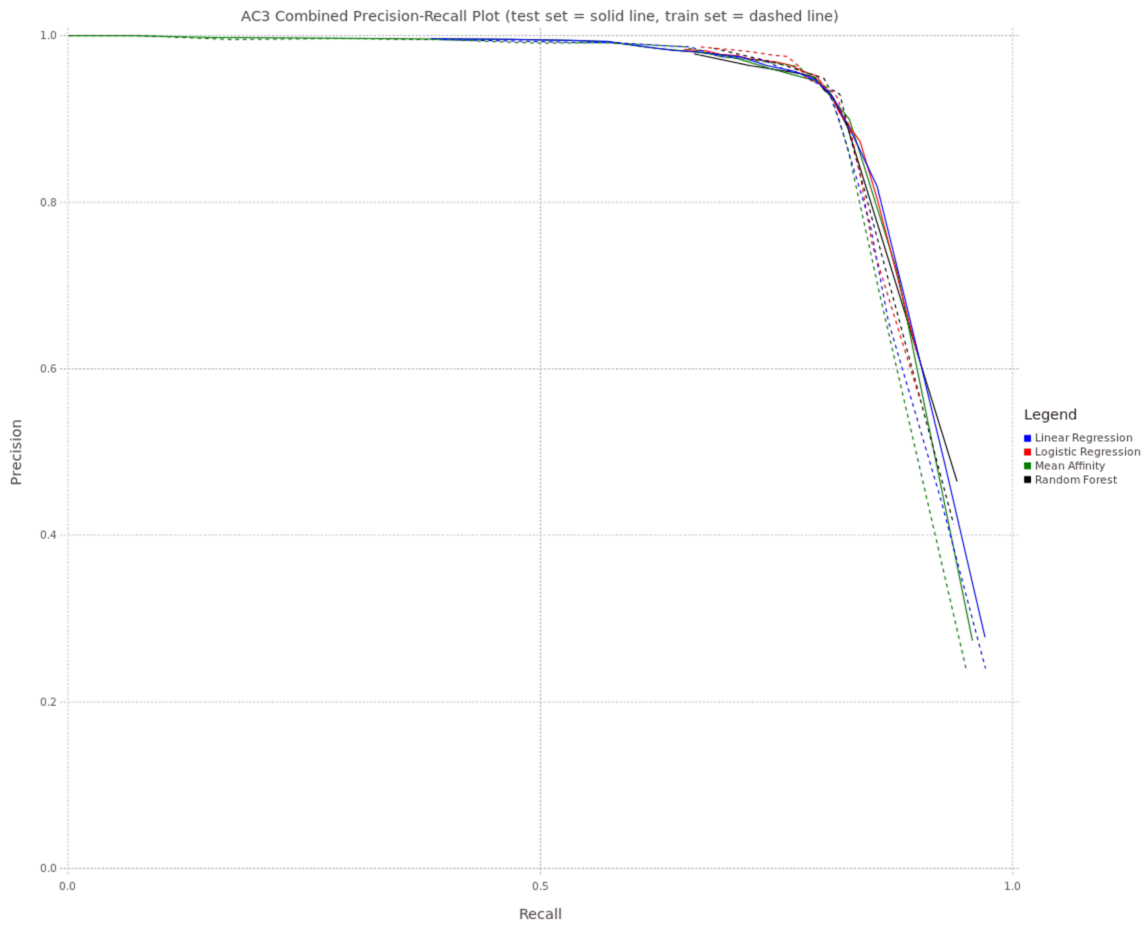

Figure 4.2: Classification precision-recall graph for the AC3 dataset

4.2 Agglomeration Precision-Recall Graphs

Figure 4.3 indicates that all methods have comparable agglomeration performance on both the SNEMI3D training and test sets. It is difficult to say that one method performs better than the others because of the curves intersecting in the graph. The supervised machine learning classifiers do not give better agglomeration performance than the mean affinity agglomeration baseline on both the training and test sets. This contrasts with the classification performance evaluation for the SNEMI3D dataset, which showed that supervised machine learning methods give better classification performance than the mean affinity agglomeration baseline on both the training and test sets.

As shown in Figure 4.4, the agglomeration performance of the supervised machine learning methods is better than that of the mean affinity agglomeration baseline on the AC3 training set while the agglomeration performance of all methods is comparable on the AC3 test set. On the AC3 training set, the agglomeration performance of all supervised machine learning classifiers is comparable. The AC3 test set agglomeration performance results are consistent with the classification performance results - all methods give similar performance. However, the AC3 training set agglomeration results are different than the classification performance results - the supervised machine learning classifiers give better agglomeration performance but the same classification performance as the mean affinity agglomeration baseline.

Figure 4.5 identifies which features are most important for the linear supervised machine learning models using the learned feature weights. The mean affinity, maximum affinity and minimum volume features were found to be the most important features. High values for the mean affinity and maximum affinity features are correlated with merge decisions for the linear models. The large positive weights for these features strongly indicate that two supervoxels with high affinities along their shared boundary are highly likely to be part of the same neuron segment. In addition, logistic regression models learn that low values for the minimum volume feature are correlated with not-merge decisions. The large negative weights for this feature strongly indicate that supervoxel pairs, where one of the supervoxels is very small, are highly likely to be part of two different neuron segments. There are small specks of dust in the EM image volumes that should not be merged so the logistic regression models are correct to penalise low values for the minimum volume feature.

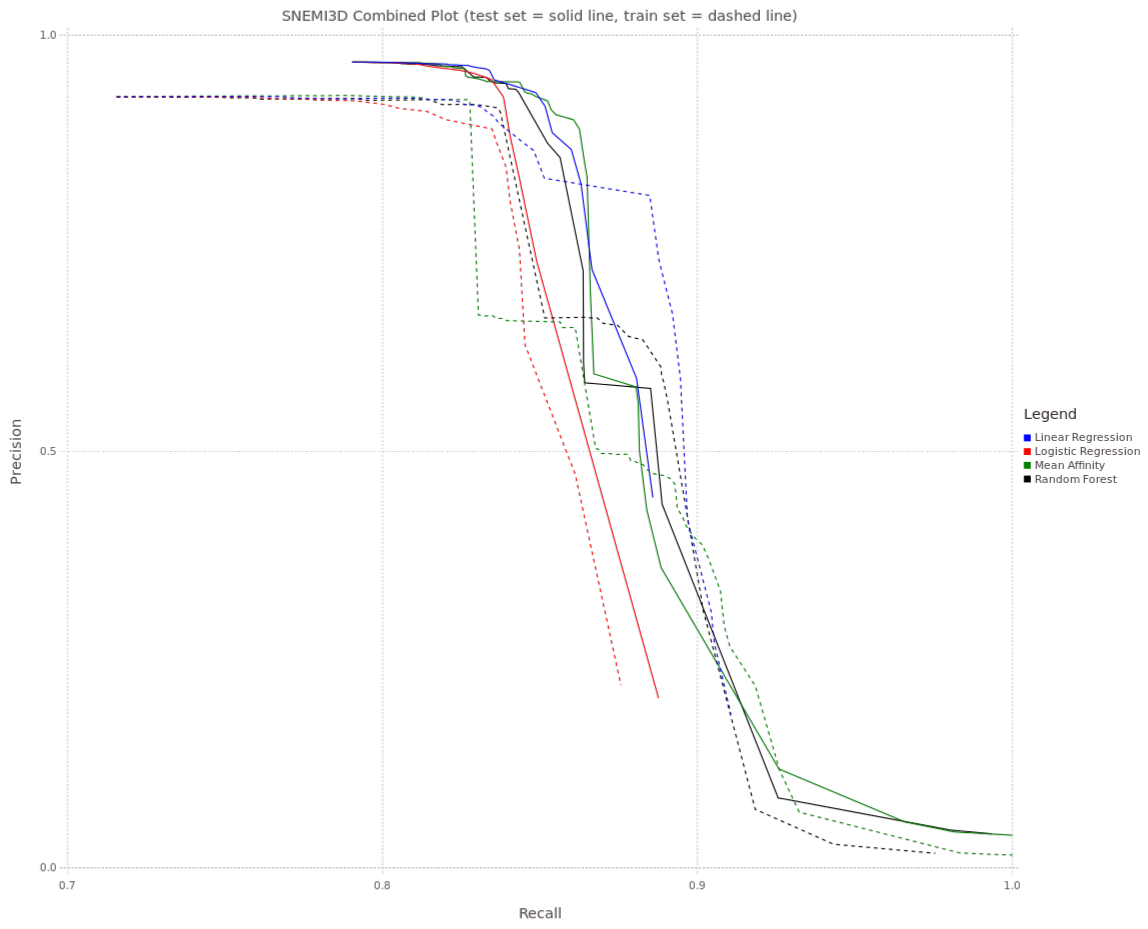

Figure 4.3: Agglomeration precision-recall graph for the SNEMI3D dataset

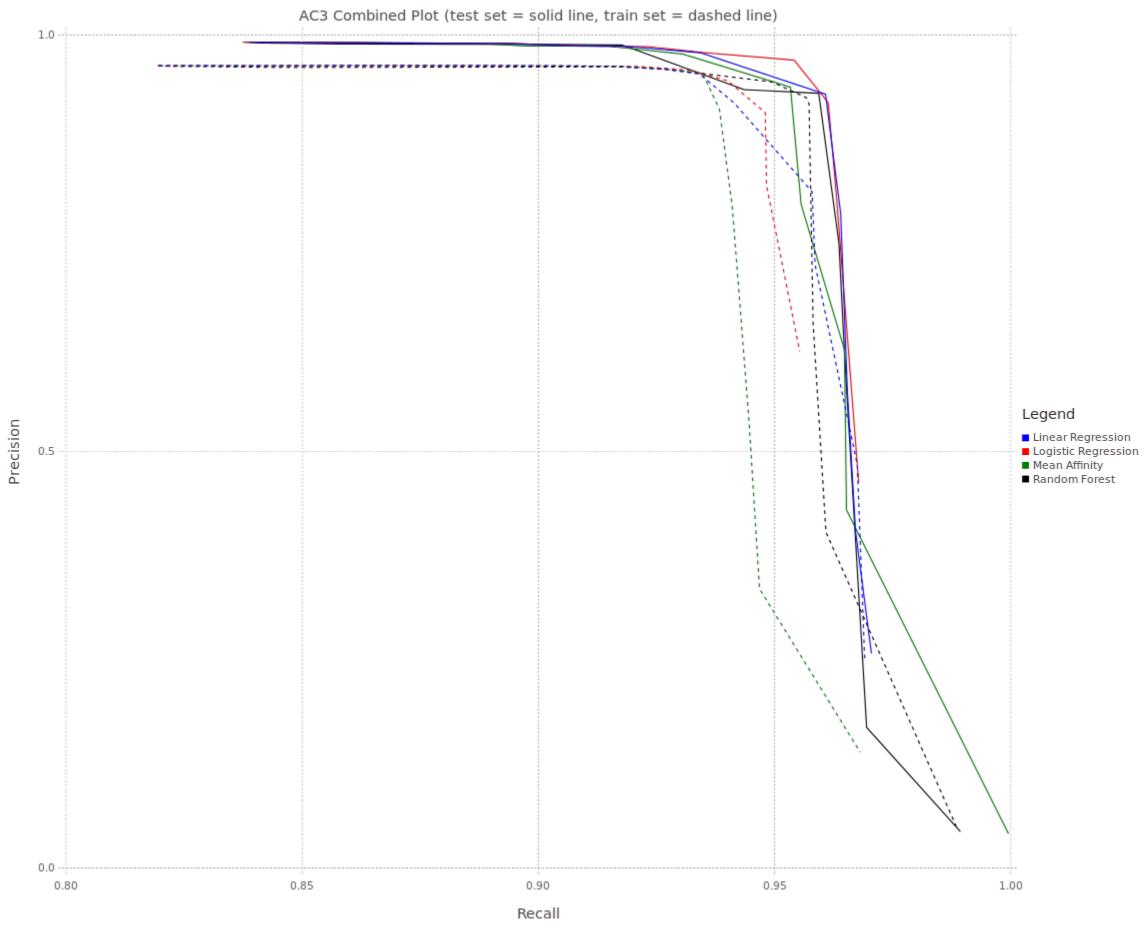

Figure 4.4: Agglomeration precision-recall graph for the AC3 dataset

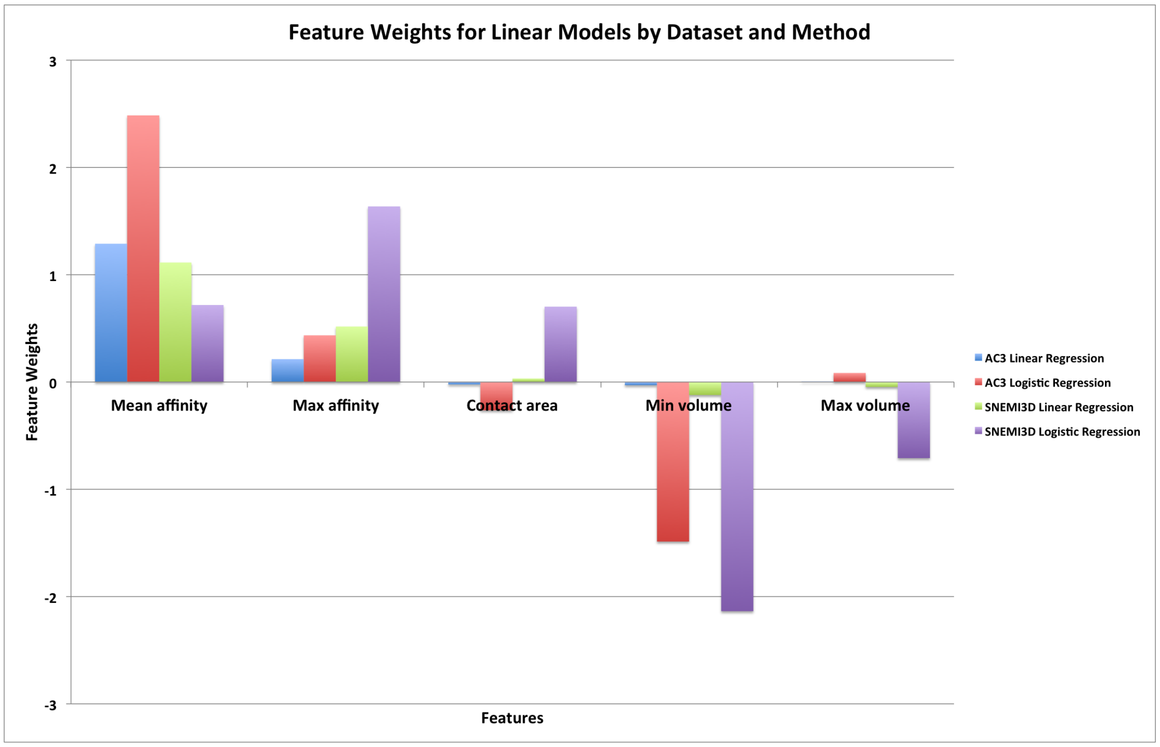

Figure 4.5: Feature weights of trained linear supervised machine learning models by dataset and method. The most important features for linear models are the mean affinity, maximum affinity and minimum volume features.

Chapter 5

Conclusion

In summary, the results were inconclusive about whether supervised machine learning methods are better than the mean affinity agglomeration baseline for the task of greedy agglomeration. With the classification evaluation framework, the supervised machine learning methods performed better than the mean affinity agglomeration baseline on both the SNEMI3D training and test sets while all methods performed comparably on both the AC3 training and test sets. With the agglomeration evaluation framework, all methods performed comparably on the SNEMI3D training and test sets. On the other hand, the supervised machine learning methods performed better than the mean affinity agglomeration baseline on only the AC3 training set but all methods performed comparably on the AC3 test set.

There is an obvious discrepancy between the classification evaluation framework results and the agglomeration evaluation framework results. This discrepancy is likely caused by the fact that the classification framework evaluates each merge/not-merge decision independently of every other decision (i.e. the sequence of merge/not-merge decisions is irrelevant) while the agglomeration framework directly evaluates the agglomeration output over time, which is strongly affected by the sequence of merge/not-merge decisions. This observation suggests that the classification evaluation framework should be used to assess how well the supervised machine learning classifiers are learning and whether or not they are overfitting to the training data while the agglomeration evaluation framework should be used to directly evaluate the performance of the methods for the task. Furthermore, it is difficult to distinguish between curves in the agglomeration performance graphs because of the noisiness inherent in the path choices made by greedy agglomeration methods. However, the agglomeration performance graphs do show high precision and high recall for all the greedy agglomeration methods despite the noisiness.

Nevertheless, the greedy agglomeration methods suffer from two kinds of errors: misalignment errors and errors where the dendritic spines are not grown out from the dendritic shaft. The misalignment errors are best described as random translation, in-plane rotation, out-of-plane rotation, and/or skew transformations that are applied at all z-slices below some z-slice in the interior of an EM image stack because of imaging issues. It is unlikely that any agglomeration approach will correct for misalignment errors because these errors are caused at the imaging stage of the pipeline and they are random. Thus this kind of error is best addressed at the imaging stage of the pipeline. The failure to grow out dendritic spines from the dendritic shafts is caused by low affinities between voxels belonging to the same dendritic spine because the dendritic spines are akin to small strands of spaghetti. The mean affinity, maximum affinity and minimum volume features were found to be the most important features for all the linear supervised machine learning methods while the other features were found to be relatively less important (see Figure 4.5). Thus it is likely that the low affinities between voxels belonging to the same dendritic spine impede the greedy agglomeration methods from successfully growing out the dendritic spines. It is possible that a deep learning and semantic segmentation approach to segmenting and predicting distinct neuron parts such as soma, axon, dendrite, synapse and glia may yield better performance than greedy agglomeration methods because it takes a more global view of the connectome (i.e. less randomness due to greedy or locally optimal merge/not-merge decisions) and automatically learns a hierarchy of features that are more useful for growing out dendritic spines than the hand-designed features used in greedy agglomeration [15].

Future work should explore better ways of evaluating agglomeration performance since the classification evaluation framework evaluates each merge/not-merge decision independently and the agglomeration evaluation framework is sensitive to the sequence of merge/not-merge decisions. Furthermore, future work should focus on improving the imaging technology used in the pipeline so that misalignment errors become less severe and/or less frequent, and on deep learning methods for semantic segmentation of EM image stacks. Advances in both aspects should alleviate the errors produced by the current version of the pipeline and hopefully lead to higher quality connectomes. These higher quality connectomes will then aid neuroscientists in making discoveries about the functioning of the brain that could not have been made otherwise.

Bibliography

[1] About Eyewire, A Game to Map the Brain. `http://blog.eyewire.org/about/`.

[2] Ignacio Arganda-Carreras, Srinivas C Turaga, Daniel R Berger, Dan Cireşan, Alessandro Giusti, Luca M Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9:142, 2015.

[3] John A Bogovic, Gary B Huang, and Viren Jain. Learned versus hand-designed feature representations for 3d agglomeration. *arXiv preprint arXiv:1312.6159*, 2013.

[4] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems*, pages 2843–2851, 2012.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[7] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[8] ISBI Challenge: Segmentation of neuronal structures in EM stacks. `http://brainiac2.mit.edu/isbi_challenge/home`.

[9] Viren Jain, Srinivas C Turaga, K Briggman, Moritz N Helmstaedter, Winfried Denk, and H Sebastian Seung. Learning to agglomerate superpixel hierarchies. In *Advances in Neural Information Processing Systems*, pages 648–656, 2011.

[10] Narayanan Kasthuri, Kenneth Jeffrey Hayworth, Daniel Raimund Berger, Richard Lee Schalek, José Angel Conchello, Seymour Knowles-Barley, Dongil Lee, Amelio Vázquez-Reina, Verena Kaynig, Thouis Raymond Jones, et al. Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661, 2015.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[12] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Kisuk Lee, Aleksandar Zlateski, Vishwanathan Ashwin, and H Sebastian Seung. Recursive training of 2D-3D convolutional networks for neuronal boundary prediction. In *Advances in Neural Information Processing Systems*, pages 3573–3581, 2015.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[17] Toufiq Parag, Anirban Chakraborty, Stephen Plaza, and Louis Scheffer. A context-aware delayed agglomeration framework for electron microscopy segmentation. *PLoS ONE*, 10(5):e0125825, 2015.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[19] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[21] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[22] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22(2):511–538, 2010.

[23] Lav R Varshney, Beth L Chen, Eric Paniagua, David H Hall, and Dmitri B Chklovskii. Structural properties of the caenorhabditis elegans neuronal network. *PLoS Computational Biology*, 7(2):e1001066, 2011.

[24] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.