TU München

Fakultät für Informatik

Machine Learning for Connectomics

Benedikt Sebastian Staffler

Vollständiger Abdruck der von der promotionsführenden Einrichtung Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr. Julien Gagneur

Prüfer der Dissertation:

1. Prof. Dr. Patrick van der Smagt
2. Prof. Dr. Bjoern Menze
3. Prof. Dr. Moritz Helmstaedter

Die Dissertation wurde am 01.08.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 23.11.2018 angenommen.

Abstract

Neuroscientific research has gained considerable insights into brain function over the past decades. However, it has not been possible yet to extract the fundamental principles of information processing in the brain to an extent that the capabilities of the brain can be artificially reproduced. While the properties of single nerve cells and their modes of signal transmission have been studied in great detail, the knowledge about the neural network formed by interconnected nerve cells is still limited for most species and brain areas. Consequently, substantial effort is currently being made to acquire and analyze wiring diagrams obtained from neural networks of organisms. Modern 3-dimensional (3D) electron microscopy allows to image large volumes of brain tissue at nanometer-scale resolution. The reconstruction of neural networks from the 3D image data requires to follow the processes of each nerve cell and to detect the locations of signal transmission between nerve cells (these signal transmission sites are called synapses). For all but the smallest neural networks manual reconstruction is prohibitively time consuming making the development of algorithms for automated reconstruction necessary.

In this thesis, algorithms for automated synapse detection and volume segmentation for 3D electron microscopy data were developed and applied for circuit reconstruction from a dataset imaged by serial block-face electron microscopy.
In the first part of this thesis, a novel synapse detection method for chemical synapses (SynEM) is proposed, which formulates the task of synapse detection as a classification of interfaces between neuronal processes. For classification, a hand-designed feature representation of interfaces is used that explicitly incorporates the local pre- and postsynaptic process. Based on the performance of single synapse detection, precision and recall rates above 97% were estimated for binary neuron-to-neuron connections.
In the second part of this thesis, deep learning approaches for image processing were used to further improve the performance of SynEM. First, the hand-designed features of SynEM were replaced with features learned by fully convolutional networks, which improved performance over the hand-designed features. In addition, the generalization performance of different classifiers to a new dataset without retraining was evaluated. Second, the volume segmentation underlying the SynEM interface definition was improved using fully convolutional networks for membrane prediction.
In the last part of this thesis, SynEM was used in combination with manual proofreading to detect all synapses onto the dendritic tree of a spiny stellate cell in layer 4 of mouse somatosensory cortex. The distribution of synapse sizes, which are correlated to the synaptic weights, is shown to be well described by a lognormal distribution. Furthermore, the distances of synapses from the soma are examined.

The methods presented in this thesis provide critical advances towards a high-throughput reconstruction of connectomes from 3D electron microscopy data. The improvements are especially relevant to make the dense reconstruction of all processes and their connections in large-scale 3D datasets feasible.

Zusammenfassung

Die neurowissenschaftliche Forschung der letzten Jahrzehnte konnte beträchtliche Erkenntnisse über die Funktionsweise des Gehirns erzielen. Trotzdem war es bisher nicht möglich, die fundamentalen Prinzipien der Informationsverarbeitung des Gehirns in einem Ausmaß abzuleiten, welches es möglich macht, die Leistungsfähigkeit des Gehirns künstlich zu reproduzieren. Während die Eigenschaften einzelner Nervenzellen und deren Mittel zur Signalübermittlung bereits in vielen Details studiert wurden, ist das Wissen über das neuronale Netzwerk aus verbundenen Nervenzellen in den meisten Spezies und Gehirnregionen noch sehr begrenzt. Infolgedessen werden derzeit beträchtliche Anstrengungen zur Messung und Analyse des Netzwerkes aus Nervenzellen im Nervensystem von Organismen unternommen. Dazu werden dreidimensionale (3D) Bilderserien von Nervengewebe im Nanometerbereich mittels Elektronenmikroskopie aufgenommen. Um die neuronalen Netzwerke aus den Bilddaten zu rekonstruieren müssen die Fortsätze der Nervenzellen kartographiert und deren Verbindungsstellen, die Synapsen, detektiert werden. Die manuelle Rekonstruktion der Bilddaten ist so aufwendig, dass damit nur kleine Netzwerke analysiert werden können, weshalb automatisierte Methoden für die Analyse großer Datensätze benötigt werden.

In dieser Dissertation werden Methoden für die automatische Detektion von Synapsen und zur Volumensegmentierung von 3D Elektronenmikroskopiedaten entwickelt und für die Analyse eines Datensatzes angewandt, der mittels serial block-face electron microscopy aufgenommen wurde.
Im ersten Teil der Dissertation wird eine neue Methode zur automatischen Detektion chemischer Synapsen (SynEM) vorgestellt. In SynEM wird die Detektion von Synapsen als Klassifikation von Berührungsflächen zwischen neuronalen Fortsätzen formuliert. Die Berührungsflächen wurden durch manuell erstellte Merkmale beschrieben, welche explizit die pre- und postsynaptischen Fortsätze mit einbeziehen. Basierend auf den Fehlerraten der Detektion einzelner Synapsen wurden Fehlerraten für das korrekte Auffinden von synaptisch verbundenen Nervenzellen von weniger als 3% abgeschätzt.
Im zweiten Teil der Dissertation werden künstliche neuronale Netzwerke zur Verbesserung der Leistung von SynEM verwendet. Als erstes wurden die manuell erstellen Merkmale von Berührungsflächen durch Merkmale ersetzt, welche von Fully Convolutional Networks gelernt wurden, was zu einer verbesserten Leistung führte. Zusätzlich wurde die Generalisierbarkeit verschiedener Klassifizierer bestimmt, indem sie ohne erneutes Training auf einem neuen Datensatz ausgewertet wurden. Als zweites wurde die Volumensegmentierung, welche die Grundlage zur Berechnung der Berührungsflächen in SynEM darstellt, verbessert, indem Fully Convolutional Networks zur Detektion von Zellmembranen verwendet wurden.
Im letzten Teil der Dissertation wird SynEM in Kombination mit manuellem Korrekturlesen verwendet, um alle Synapsen zu detektieren, welche die Dendriten einer Sternzelle in Schicht 4 des primären somatosensorischen Kortex einer Maus innervieren. Es wird gezeigt, dass sich die Verteilung der Größen von Synapsen, welche zu den Stärken von Synapsen korreliert sind, durch eine Logarithmische Normalverteilung beschreiben lässt. Zudem wird die Position von Synapsen entlang der Dendriten untersucht.

Die Methoden, welche in dieser Dissertation dargelegt werden, stellen wichtige Fortschritte auf dem Weg zur automatisierten Rekonstruktion von Konnektomen aus 3D Elektronenmikroskopiedaten dar. Die Verbesserungen sind insbesondere für die dichte Analyse aller Nervenzellen und deren Verbindungen in großen Datensätzen relevant, welche dadurch erst realisierbar werden.

Acknowledgements

First and foremost, I would like to thank Moritz Helmstaedter for conceiving and supervising the project as well as Patrick van der Smagt for supervision and excellent support and advice. Furthermore, I would like to thank Tobias Rose for additional supervision.

I thank Manuel Berning, Kevin Boergens and Anjali Gour for the successful collaboration, many discussions and comments on the manuscript.

I thank my colleagues Emmanuel Klinger, Alessandro Motta, Sahil Loomba, Philipp Bastians, Jakob Straehle, Florian Drawitsch, Ali Karimi, Yungfeng Hua, Philip Laserstein, Martin Schmidt, Christian Schramm, Thomas Kipf, Meike Schurr, Marcel Beining, Kun Song and Helene Schmidt from the Max Planck Institute for Brain Research as well as my colleagues Agneta Gustus, Grady Jensen, Sebastian Urban, Justin Bayer, Christian Osendorfer, Markus Kühne, Nutan Chen, Maximilian Karl and Maximilian Sölch from the group of biomimetic robotics and machine learning at the Technical University of Munich for many fruitful discussions, comments on manuscripts and a constructive working atmosphere.

Finally, I would like to thank my family and in particular my girlfriend Jenny Pfeiffer for all support and patience during the work on this thesis.

Contents

1. Introduction

The nervous system of animals is the central unit of control that integrates, processes and distributes information from different parts of the body. In vertebrates, the nervous system is classified into two main parts, the central nervous system consisting of the brain and the spinal cord and the peripheral nervous system consisting of the nerve tissue outside the central nervous system. In particular the brain and its largest part, the cerebral cortex, play a key role in higher cognitive functions including learning, memory, emotion and decision making (Kandel et al., 2000). However, many functional principles of the brain are still poorly understood both from an algorithmic (Lake et al., 2017; Hassabis et al., 2017) as well as from a medical perspective (Fornito et al., 2015).

1.1. Neurons and Neuronal Networks

On a microscopic level, it has widely been accepted that the nervous system consists of a specialized cell type called neuron (neuron doctrine; based on Golgi, 1873; y Cajal, 1888). Neurons can transmit electrical signals along thin filaments emanating from the cell body, which are called neuronal processes or neurites. The neuronal processes contain synapses, specialized structures which emit electrical or chemical signals to contact other neurons forming a (biological) neural network of interconnected cells. The complexity of the brain compared to other organs arises from the multitude of subtypes of neurons and, maybe even more importantly, from the high degree of local and long range communication between neurons (Lichtman and Denk, 2011). To understand how the brain performs its computations it is thus necessary to study both the single neurons comprising the network as well as their connections. The analysis of single neurons includes their morphological, physiological and functional properties, which determine each neurons' response properties to stimuli, as well as their means and modes of signal transmission. The study of connections between neurons is the focus of the neuroscientific field of connectomics.

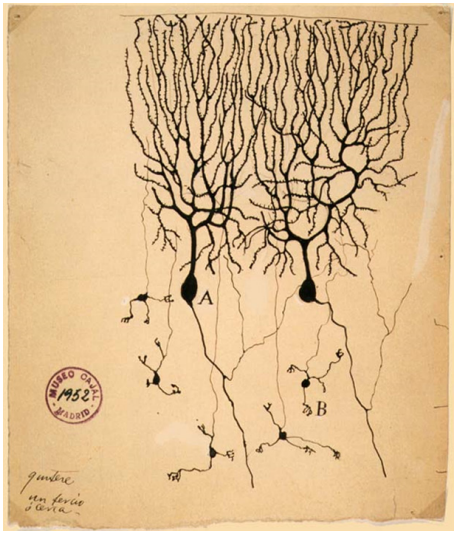

Figure 1.1: Neuron drawings. Drawing of Purkinje cells (A) and granule cells (B) from pigeon cerebellum by Santiago Ramón y Cajal, 1899. Instituto Santiago Ramón y Cajal, Madrid, Spain. (Wikimedia Commons: https://commons.wikimedia.org/wiki/File:PurkinjeCell.jpg)

The goal of connectomics is to generate a comprehensive map of connections between neurons or populations of neurons, which is called a wiring diagram or connectome (Sporns et al., 2005). On the macroscale, connectomics investigates the long-range interaction between brain regions for example using diffusion tensor imaging with a typical resolution larger than a cubic millimeter. A mesoscale connectome encompasses the connections between populations of a few hundred neurons. The generation of a mesoscale connectome has already been achieved for the whole mouse brain (Oh et al., 2014). This thesis focuses on microscale or nanoscale connectomics that maps neural circuits on the scale of single neurons and single synapses. Nanoscale connectomics necessarily requires datasets with a resolution that is high enough to identify synapses and to follow every neuronal process in each direction back to its soma. In mouse neocortex, the minimal required resolution is on the order of $30\,\mathrm{nm}$ while it can be even lower for other circuits, for example in the fruit fly (Helmstaedter, 2013). Conventional light microscopic techniques are limited in their resolution to a few hundred nanometers due to the diffraction limit rendering this technique insufficient for nanoscale connectomics even when using two-photon excitation or confocal detection (Lichtman and Denk, 2011). Modern super-resolution techniques, however, are able to overcome the diffraction barrier and achieve resolutions of a few dozen nanometers (Rust et al., 2006), but still have to overcome other problems such as the low penetration depth into tissue and the labeling density (Lichtman and Denk, 2011; Lakadamyali et al., 2012; Ke et al., 2016). In addition to the resolution requirement, the imaged volume needs to be large enough to contain the neural circuit of interest. While the cell bodies of neurons are not particularly large compared to other cells, the neuronal processes can extend to volumes that are 3-4 orders of magnitudes larger than the cell body with a total path length on the order of centimeters for mice and even meters for humans (Lichtman and Denk, 2011). Thus, to study even local cortical circuits from mouse neocortex requires datasets with several hundred micrometers edge length (Helmstaedter, 2013). Currently, electron microscopy (EM) (Knoll and Ruska, 1932) is the only imaging technique that is both able to provide the high resolution requirement and the possibility to image datasets that are large enough to contain considerable parts of local neural circuits. Modern volume EM techniques allow for the acquisition of a cubic dataset with an edge length of $300\,\mu\mathrm{m}$ in about 1000 hours (Helmstaedter, 2013). Methodological advances promise further speed-up of data acquisition, for example Eberle et al. (2015) increased imaging speed by a factor close to two orders of magnitude by imaging a sample with multiple electron beams.

1.2. The Connectomics Reconstruction Challenge

Datasets produced by modern EM techniques easily range into the petabyte scale and can contain thousands of neurons and glia cells as well as a diverse set of ultrastructure with similar visual appearance. A dataset of size $250 \times 250 \times 250\,\mu\mathrm{m}^3$ from mouse neocortex contains roughly 2500 neurons, which have neurites with an overall path length of approximately 81 meters forming about 15.5 million synapses (Figure 1.2a, b; White and Peters, 1993; Braitenberg and Schuez, 1991; Merchán-Pérez et al., 2014). The reconstruction of a neural circuit requires the identification of all synapses and the corresponding pre- and postsynaptic processes have to be traced back to their respective somata, which is called neurite or cable reconstruction. The task of synapse

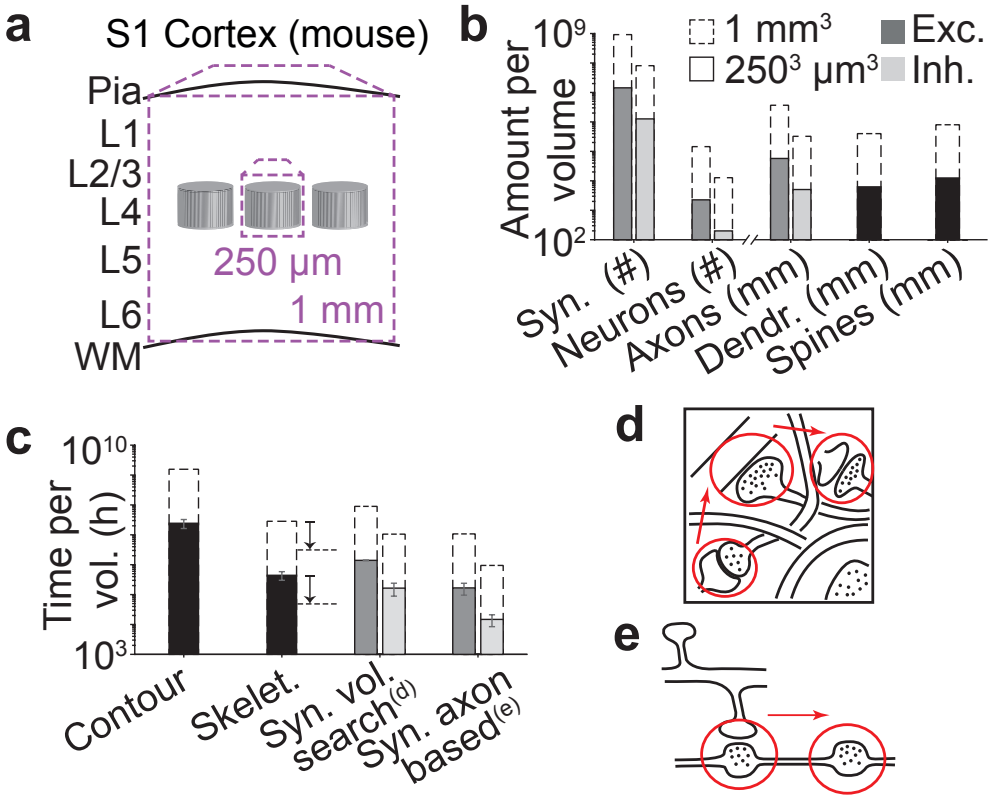

Figure 1.2: The connectomics reconstruction challenge. (a) Connectomic reconstruction in mouse somatosensory cortex (S1) requires a minimal dataset size with an edge length of at least 250 µm for a local circuit ('barrel') from layer 4 (L4) and a dataset with an edge length of 1 mm to cover the whole cortical depth. (b) Expected number of synapses and neurons as well as the expected path length of axons, dendrites and spines for the two example datasets in (a) (White and Peters, 1993; Braitenberg and Schuez, 1991; Merchán-Pérez et al., 2014). (c) Estimated reconstruction time for neuronal processes by contouring (Contour) and skeletonization (Skelet.) and for manual synapse detection by volume search (Syn. vol. search., see d) and axon-based search (Syn. axon based, see e). Arrows: WebKnossos (Boergens et al., 2017) allows for a further speed-up of skeletonization of 5 to 10-fold. (d) Synapse detection by volume search requires scanning the 3D image stacks while keeping track of already inspected locations taking about 0.1 h/mm. (e) Axon-based synapse detection by inspection of boutons for previously reconstructed axons taking about 1 minute per bouton. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 1 https://doi.org/10.7554/eLife.26414.003 licensed under CC BY 4.0)

detection consists of locating all synapses based on their ultrastructural appearance as well as identifying the corresponding pre- and postsynaptic partner. Note that, depending on the kind of tissue, a single synapse can have multiple postsynaptic partners, for example in the mammalian retina or the fly optical system (Helmstaedter, 2013). Once the pre- and postsynaptic partners have been identified, they need to be associated with the neurons in the dataset they belong to. This is not only difficult because neurites can become very small and intertwined but mainly because errors along the path of neurites can have correlated effects, for example the wrong assignment of a neurite branch causes all the synapses and thus all connections along the branch to be wrong as well (Helmstaedter, 2013). Since the neurites of a single neuron can extend over several millimeters to meters of path length, many small errors can lead to a macroscopically altered and even biologically implausible neuron reconstruction, for example a neuronal process connecting two somata, that can potentially falsify conclusions drawn from the connectome.

The manual reconstruction of all but the smallest neural circuits is typically prohibitively expensive (Figure 1.2c). Manually creating a volume segmentation by contouring of processes, i.e. a

grouping of all voxels that belong to the same process by outlining the border of the process, takes about 2800 years of continuous work for a single person for 81 meters of neuronal wiring at a speed of 300 hours for a millimeter of neurite path length (Helmstaedter et al., 2011). A significant speed-up can be achieved by following only the center lines of processes, an approach called skeletonization. Reconstruction by skeletonization results in a 50-fold reduction of manual annotation time over contouring to about 50 years (Helmstaedter et al., 2011). Further improvements in data visualization and delivery reduced skeletonization time by another factor of roughly 10 to about 5.5 years (Boergens et al., 2017). Manual synapse detection by volume search requires scanning the 3D data for synapse locations while keeping track of the already annotated synapses taking an additional 180 years (Staffler et al., 2017a). Assuming that the axons are already reconstructed, for example if the skeletonization approach for neurite reconstruction was taken, an annotation strategy focused only on the boutons of axons can further reduce synapse detection time to about 21 years (Staffler et al., 2017a). This highlights the necessity of automated methods for connectomic reconstruction.

The automation of circuit reconstruction for connectomics faces several computational challenges (Lichtman et al., 2014). Even for manual reconstruction, the large amounts of image data produced by modern 3D EM techniques require a computational infrastructure that allows to efficiently store and retrieve the data from arbitrary locations for visualization and analysis, for example by exploiting the spatial locality of the 3D data (Saalfeld et al., 2009; Boergens et al., 2017). Due to the sequential 2D nature of the EM imaging potentially involving several tiles per imaging plane, a necessary first step for every analysis is the alignment of image tiles and consecutive sections into a single coherent 3D image volume. Proper image alignment is crucial, in particular for automated methods, and can affect all subsequent data analysis steps. Despite advances in automatic image alignment and registration (Kaynig et al., 2010; Saalfeld et al., 2012; Yoo et al., 2017), imaging artifacts often require manual intervention in the alignment process. Neurite reconstruction is often formulated as the computer vision task of image segmentation. However, due to the correlation of errors along processes, algorithms for neurite reconstruction require very high accuracy to avoid false reconstructions (Helmstaedter, 2015). Even though current automated segmentations often have error rates that are too high to faithfully reconstruct whole processes, they can still be used to provide the local shape of processes in conjunction with manual skeleton tracings which provide the long-range connections (Berning et al., 2015). Similarly, synapse detection can be formulated as a segmentation problem as well, however, the detection of pre- and postsynaptic partners often requires additional steps. The design of algorithms for the analysis of 3D EM data faces several big data challenges such as the data size and the computational complexity (Lichtman et al., 2014) and typically requires domain expertise. Machine learning algorithms can learn to solve data analysis tasks based on examples and are often designed to be highly parallel and scalable with a performance that is often superior to human-designed algorithms and are thus heavily used for connectomics reconstruction.

1.3. Insights from Structural Data

Generating a connectome using EM-based connectomics poses challenges both in terms of image acquisition as well as from a data analysis perspective and requires substantial amounts of resources. The first full connectome comprising 302 neurons and roughly 7000 connections was generated for the nematode *Caenorhabditis elegans* (White et al., 1986) and required more than ten years of manual reconstruction time. More recent reconstructions in the mouse retina (Helmstaedter et al., 2013) or the drosophila visual system (Takemura et al., 2013) still required substantial human labor of tens of thousands of working hours. This raises the question what actually can be learned from generating a connectome for the brain of an organism. A lot of criticism against the nanoscale connectomics approach has been voiced ranging from potentially unnecessary (Sporns et al., 2005) to the impossibility of reading connectomes (Bargmann, 2012) (see also Morgan and Lichtman, 2013 for some common arguments against connectomics). One of the most obvious concerns is that the brain is highly variable on the synaptic level but the connectome can only provide a snapshot of the connections at one specific point in time of a single animal (Sporns et al., 2005). It also has been pointed out that brain circuits are subject to neuromodulation, which is not captured in EM datasets (Bargmann and Marder, 2013). However, many scientists believe that nevertheless a lot can be learned even from a single connectome about fundamental wiring principles that are present across time and individuals (Lichtman and Denk, 2011). At the very least, connectomics should be able to constrain computational models of the brain and potentially falsify them (Denk et al., 2012). It might even be possible that connectomics provides the means of reading out memory, since neuroscientists have hypothesized for long that memories are stored in the patterns of synaptic connections of neurons and their strength (Seung, 2009). Beyond the hypothetical benefits of a dense wiring diagram, the field of connectomics has already provided insights into the nervous system that would have been difficult or even impossible to obtain by other means (see also Table 1 in Kornfeld and Denk, 2018). Mishchenko et al. (2010) and Kasthuri et al. (2015) refuted subcellular versions of Peter's rule, which states that spatial proximity of axons and dendrites can predict a connection or even the number of synapses between processes (see also Rees et al., 2017). Briggman et al. (2011) showed how a wiring asymmetry could contribute to the direction selectivity of retinal ganglion cells. Helmstaedter et al. (2013) identified a new cell type in the retina based on its connectivity. Schmidt et al. (2017) showed a sorting of synapses along the axons of excitatory neurons based on the identity of the postsynaptic neuron. Kornfeld et al. (2017) showed the existence of a neural circuit that could explain a synaptic-chain model. A general conclusion from these studies seems to be that wiring diagrams are often necessary but not sufficient to understand a neural system motivating the combination of purely structural connectivity analysis with other measurements, for example functional activity.

1.4. Research Questions and Contributions

The gap between imaging and reconstruction time and the promising insights from previous connectomic studies have spurred substantial development for automated solutions for connectomic reconstruction. In this thesis, I address and contribute to the following research questions.

Research question 1: Can existing synapse detection approaches be adapted to medium-resolution data obtained using serial block-face electron microscopy (SBEM)?

In the first part of chapter 3 and Staffler et al. (2017a,b), SynEM, a novel approach to synapse detection is proposed that formulates synapse detection as binary classification of interfaces between segments from a volume segmentation. A feature representation of interfaces is developed based on their textural and geometrical properties. For classification, a Machine Learning approach is used to infer the decision rules from training data. SynEM is evaluated on two EM datasets obtained using different EM imaging modalities.

Research question 2: How do error rates in synapse detection affect the error rates of neuron-to-neuron connections?

In subsection 3.3.4 and Staffler et al. (2017a,b), a statistical estimate for connectome error rates is developed based on synapse detection error rates. The estimate only requires the distribution of synapses between connected neurons and the overall connectivity ratio between neurons, which are available for different cell types in rodent somatosensory cortex.

Research question 3: Can features learned by artificial neural networks improve upon commonly used hand-designed feature representations for the task of synapse detection?

In section 4.3, the output of fully convolutional networks (FCNs) trained on a semantic segmentation task is used to modify the SynEM feature representation. Different feature representations are compared in subsection 4.3.2 showing an improved performance of learned features over hand-designed features. The best performance was achieved with a combination of learned and hand-designed features.

Research question 4: How well does SynEM generalize to novel EM datasets without retraining?

In the second part of subsection 4.3.2, the performance of SynEM and several interface classifiers using features learned by FCNs is evaluated on a novel dataset without any retraining or fine-tuning. All classifiers were able to provide reasonable generalization performance with a maximal drop of 5.4% in F1 score. The best result was achieved by the SynEM classifier using only hand-designed features.

Research question 5: Can the volume segmentation provided by SegEM underlying the interface definition of SynEM be further improved using artificial neural networks?

In section 4.4, FCNs were trained to predict membranes between cellular processes followed by a watershed segmentation step. A novel multi-resolution architecture is introduced and compared to other architectures. The networks are evaluated on the SegEM challenge (Berning et al., 2015) showing a substantial improvement over the performance of SegEM.

Research question 6: How can the developed methods be used to analyze wiring principles in the brain? In particular, what can be learned about the input strength distribution of all synapses innervating single cells?

In chapter 5, SynEM is applied in combination with manual proofreading to detect all synapses onto a spiny stellate neuron in a dataset from layer 4 of mouse somatosensory cortex. The distribution of synapse size, which is correlated to synapse strength, are shown to be well described by a lognormal distribution. The distribution of distances of synapses from the soma showed a shift to more proximal locations for shaft synapses then for spine synapses.

The relevant manuscripts that were published during the course of this thesis are

- Staffler, B., Berning, M., Boergens, K. M., Gour, A., van der Smagt, P., and Helmstaedter, M. (2017). SynEM: Automated synapse detection for connectomics, bioRxiv, doi:10.1101/099994
- Staffler, B., Berning, M., Boergens, K. M., Gour, A., van der Smagt, P., and Helmstaedter, M. (2017). SynEM, automated synapse detection for connectomics, eLife, doi:10.7554/eLife.26414

The following manuscript is not directly addressed in this thesis

- Parag, T., Berger, D., Kamentsky, L., Staffler, B., Wei, D., Helmstaedter, M., Lichtman, J. W. & Pfister, H. (2018). Detecting Synapse Location and Connectivity by Signed Proximity Estimation and Pruning with Deep Nets. arXiv preprint arXiv:1807.02739.

2. Background

This chapter provides the necessary background on neuroscience, computer vision and machine learning that will be used in the main chapters of this thesis. The chapter is aimed at making this thesis self-contained and can be skipped if the reader is familiar with the subject. Note that the sections in this chapter will not cover the corresponding topics exhaustively but only to the degree that will be required later on.

2.1. Structural Neurobiology and Connectomics

This section introduces the neuroscientific background and terms that are used in the main part of the thesis. The focus of this chapter is on circuit components of biological neural networks and their appearance in the 3D electron microscopy (EM) data used for circuit reconstruction.

2.1.1. Introduction to Neural Circuits

The nervous system consists of two kinds of cells, neurons and glia. The human brain for example contains roughly 85 billion neurons and the same amount of glia cells (Azevedo et al., 2009). Neurons are capable of communicating in a fast and precise way with other neurons. Glia cells have a range of functions, e.g. in brain metabolism, providing insulating sheaths called myelin for neurons that enable fast conduction of electrical signals and recent findings suggest an active participation in information coding and signaling (Perea et al., 2014). While glia cells are an important part of neural circuits, the focus of this thesis is on neurons and their connections. The following description is based on Kandel et al. (2000).

The intracellular and intercellular signaling between neurons is enabled by their morphological, chemical and electrical properties. Neurons have a cell body (soma) containing the nucleus and two kind of filaments called neurites emanating from the soma: an axon that is used to transmit signals to other neurons and dendrites that mainly receive input from other neurons (Figure 2.1 and Figure 2.2a). The neurites can extend over long distances up to centimeters in mouse and meters in human potentially covering volumes that are several orders of magnitude larger than the cell body (Lichtman and Denk, 2011). Along their path, axons typically form varicosities called presynaptic terminals or synaptic boutons that are filled with spherical structures called synaptic vesicles (Figure 2.2b). The synaptic vesicles belong to synapses at the bouton, a structure that enables chemical signal transmission to other neurons. Axons can partly be surrounded by an insulating sheath formed by glia cells called myelin (Figure 2.2c). Dendrites form tree-like morphologies that are typically highly branching. The entirety of dendritic processes of a neuron is called the dendritic tree. Some dendrites contain thin processes along their main branches called (dendritic) spines that are up to several micrometers in length and often receive synaptic input from other neurons (Figure 2.2d). Spines consist of a typically very thin spine neck that can expand at the tip of the spine into the spine head. The contents of the cell enclosed

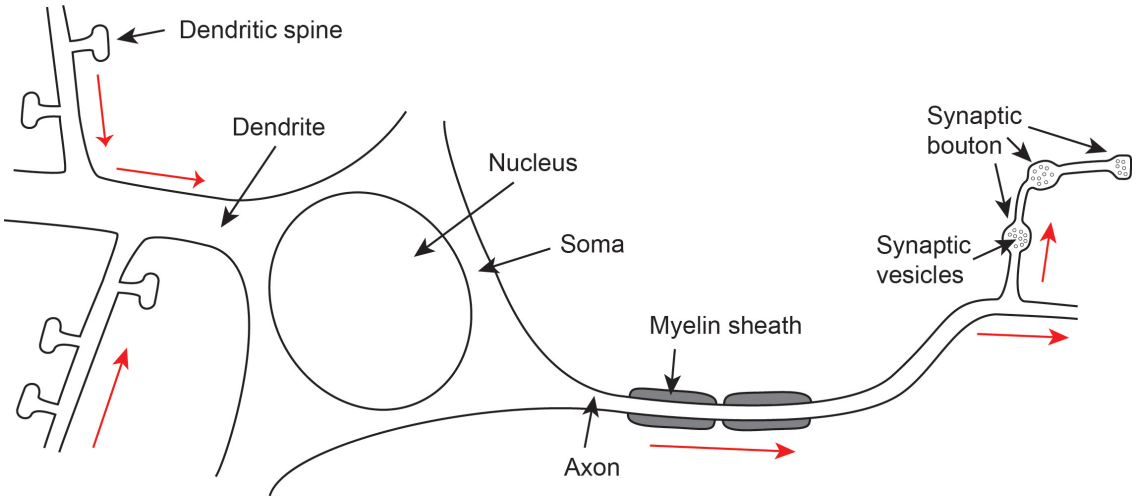

Figure 2.1: Schematic representation of the main structural components of a neuron. See subsection 2.1.1 for details. The main information flow along the neuron is indicated by red arrows.

by the neuronal membrane except for the nucleus are called cytoplasm and comprise the liquid cytosol and substructures of the cell called organelles. Examples of organelles that are frequently encountered in EM are mitochondria which are important for the cellular metabolism and the spine apparatus which occurs in some spines and that, among other things, is involved in local protein synthesis and trafficking (Figure 2.2d, e).

The intracellular signaling of a neuron is achieved by electrical currents that are conducted along the neurites. The signal conduction in dendrites is mainly passive, i.e. a local change in the electrical potential results in a spread of charged particles from the site of depolarization in both directions along the dendrite but the magnitude of the polarization diminishes with increasing distance. For signal transmission, however, neurons are able to actively propagate an electrical signal along their axons that does not diminish as it is conducted along the axon. Similar to other cells, neurons create a difference in electrical potential between the exterior and the interior of the cell called membrane potential by active and passive ion transport through the membrane typically resulting in a negative voltage in the interior of the cell. A depolarization of the membrane beyond a threshold causes a rapid reversal of the membrane potential called an action potential or spike that propagates along the axon by depolarizing neighboring membrane patches. The generation of an action potential typically happens at the soma if the sum of incoming signals is high enough to cause a depolarization beyond the spike initiation threshold.

For intercellular signaling, neurons connect to each other by transmitting chemical or electrical signals at specialized structures called synapses in a process called synaptic transmission typically contacting 1000 to 10000 other neurons (Braitenberg and Schuez, 1991, chapter 6; Murre and Sturdy, 1995). There are two different kinds of synapses with different means of conveying signals. Electrical synapses, also called gap junctions, provide a direct electrical coupling between two neurons allowing ions to flow between the two neurons in both directions (Robertson, 1953, 1963; Revel and Karnovsky, 1967). Chemical synapses, which are simply referred to as synapses if the context is clear, release chemical substances called neurotransmitters from the

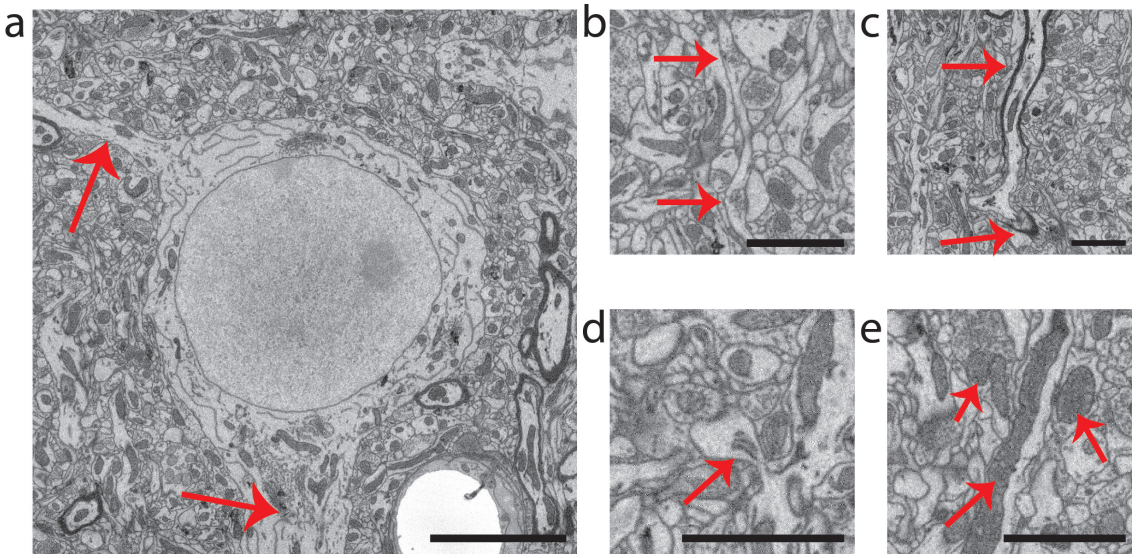

Figure 2.2: Neuronal ultrastructure in EM. Neuronal ultrastructure in EM images from mouse primary somatosensory cortex (S1) layer 4 (L4) (dataset 2012-09-28_ex145_07x2; Boergens and Helmstaedter, 2012b) at a pixel size of $11.24 \times 11.24\,\mathrm{nm}^2$ imaged using serial block-face electron microscopy (SBEM) (see subsection 2.1.2). (a) Soma of a neuron containing the nucleus and two processes emanating from it (red arrows). The volume around the soma is densely filled with neuronal processes from other neurons and a blood vessel at the bottom right with an almost white texture. (b) Axon forming two boutons along its path (red arrows) each containing synaptic vesicles and a synapse to a neighboring process. Note that at this resolution single vesicles are not clearly visible but only the entirety of the vesicle agglomeration in the bouton. (c) Axon surrounded by a myelin sheath (red arrows) that briefly ends where the axon branches but starts again for both branches (only right branch visible here). (d) Short dendritic spine containing a spine apparatus (red arrow) and a potential synaptic innervation from above. (e) Mitochondria in different processes cut at different angles (three mitochondria are marked by red arrows). Scale bars: $5\,\mu\mathrm{m}$ in (a) and $2\,\mu\mathrm{m}$ in (b-e).

sending neuron (also called presynaptic neuron) that elicit an electrical current in the receiving neuron (also called postsynaptic neuron). The release of neurotransmitter is typically triggered by a presynaptic depolarization, for example from an action potential that travels along an axon. Connections made by chemical synapses as described here thus only transmit a unidirectional signal and hence represent an asymmetric connection between neurons. If the current elicited in the postsynaptic neuron causes a depolarization of the membrane potential bringing it closer towards the threshold for spike initiation, the synapse is called excitatory and the current that can be measured in the postsynaptic neuron is called an excitatory postsynaptic potential (EPSP). Conversely, if the current elicited in the postsynaptic neuron causes a hyperpolarization that makes spike initiation less likely, the synapse is called inhibitory and the potential in the postsynaptic neuron is called an inhibitory postsynaptic potential (IPSP). The invention of EM (Knoll and Ruska, 1932) allowed neuroscientists to study the ultrastructure of the brain (Palay and Palade, 1955) and its synapses (Robertson, 1953; Palay, 1956; Gray, 1959; Colonnier, 1968) in detail (see also Peters and Palay, 1996; Harris and Weinberg, 2012). Presynaptically, synaptic vesicles located in axonal boutons contain the neurotransmitter. A single bouton can be the site of multiple synapses to different postsynaptic partners. For synaptic transmission, vesicles dock to the plasma-membrane at the synapse location called the active zone and release the neurotransmitter into the synaptic cleft between pre- and postsynaptic neuron, which is about $20\,\mathrm{nm}$ to $40\,\mathrm{nm}$ wide (Kandel et al., 2000, chapter 8). The neurotransmitter binds to receptor proteins within the plasma-membrane of the postsynaptic neuron causing a local change of the electrical potential in the postsynaptic neuron. The high protein density at the membrane of the postsynaptic neuron can typically be

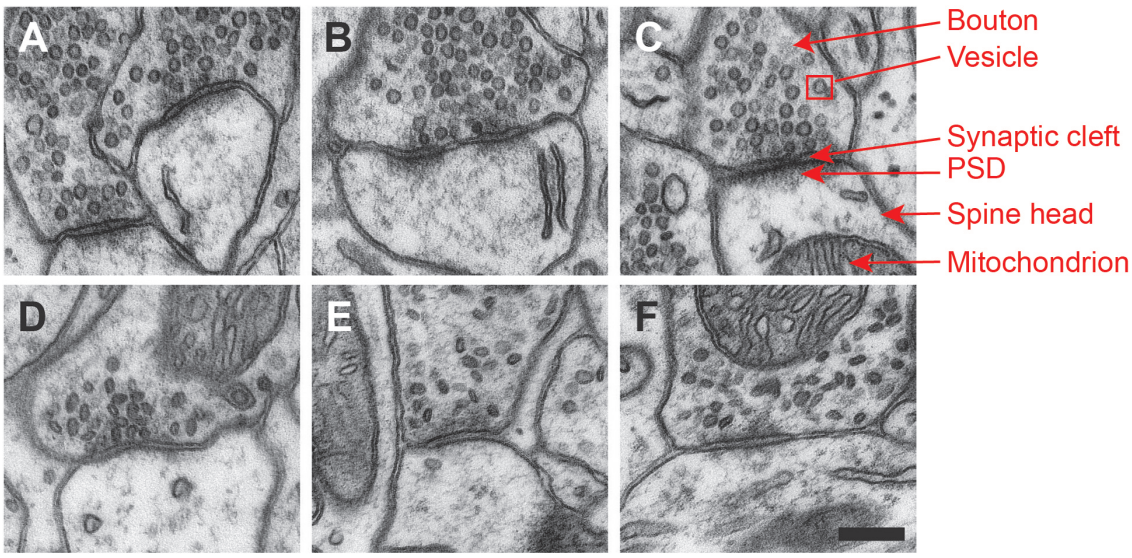

Figure 2.3: Examples of synapses from mouse neocortex in EM. Examples of excitatory (glutamateric) synapses (A, B, C) onto dendritic spines and presumed inhibitory (GABAergic) synapses (D, E, F) onto dendritic shafts. Excitatory synapses show round clear vesicles while inhibitory synapses show flattened dark vesicles and a less pronounced PSD. Scale bar: 200 nm. (Korogod, Petersen, Knott, 2015, eLife, http://dx.doi.org/10.7554/eLife.05793.013 licensed under CC BY 4.0 / Additional textural descriptions (red) were added to the original)

seen in EM images as a thickening or blurring of the postsynaptic membrane, in particular for excitatory synapses, and is called postsynaptic density (PSD) (Figure 2.3). Synapses are classified based on their ultrastructural appearance into Gray type I or asymmetric, which were shown to be mostly glutamatergic, and excitatory and Gray type II or symmetric synapses, which are mostly GABAergic and inhibitory (Gray, 1959; Colonnier, 1968; Peters and Palay, 1996; Petilla Interneuron Nomenclature Group et al., 2008). The likelihood of a single action potential triggering synaptic transmission that leads to a signal in the postsynaptic cell is called the synaptic reliability. The efficacy or strength of a synapse refers to the mean amplitude of the postsynaptic current after an action potential in the presynaptic neuron which thus depends on both the synaptic reliability as well as the magnitude of the postsynaptic current (Kandel et al., 2000, chapter 12). While this is a functional definition of synapse strength, many structural properties of synapses like the area of the PSD, the area of the presynaptic active zone or the number of docked presynaptic vesicles are highly correlated with the presynaptic release property and the efficacy of synapses thus serving as structural correlates for synapse strength (Bartol Jr et al., 2015). There are less specific forms of chemical signal transmission between neurons, e.g. by diffusion of neurotransmitters (cf. Vizi, 2000) also called volume transmission, often considered in the context of neuromodulation (Marder, 2012; Bucher and Marder, 2013) which are difficult to access in current connectomic EM datasets and will not be considered here.

2.1.2. Volume Electron Microscopy

EM is currently the only imaging technique that is both able to provide the high resolution requirement and the possibility to image datasets that are large enough to contain substantial parts of neural circuits. To capture the three dimensional structure of brain tissue, EM imaging techniques require to take two dimensional images at different depths of a tissue sample that are

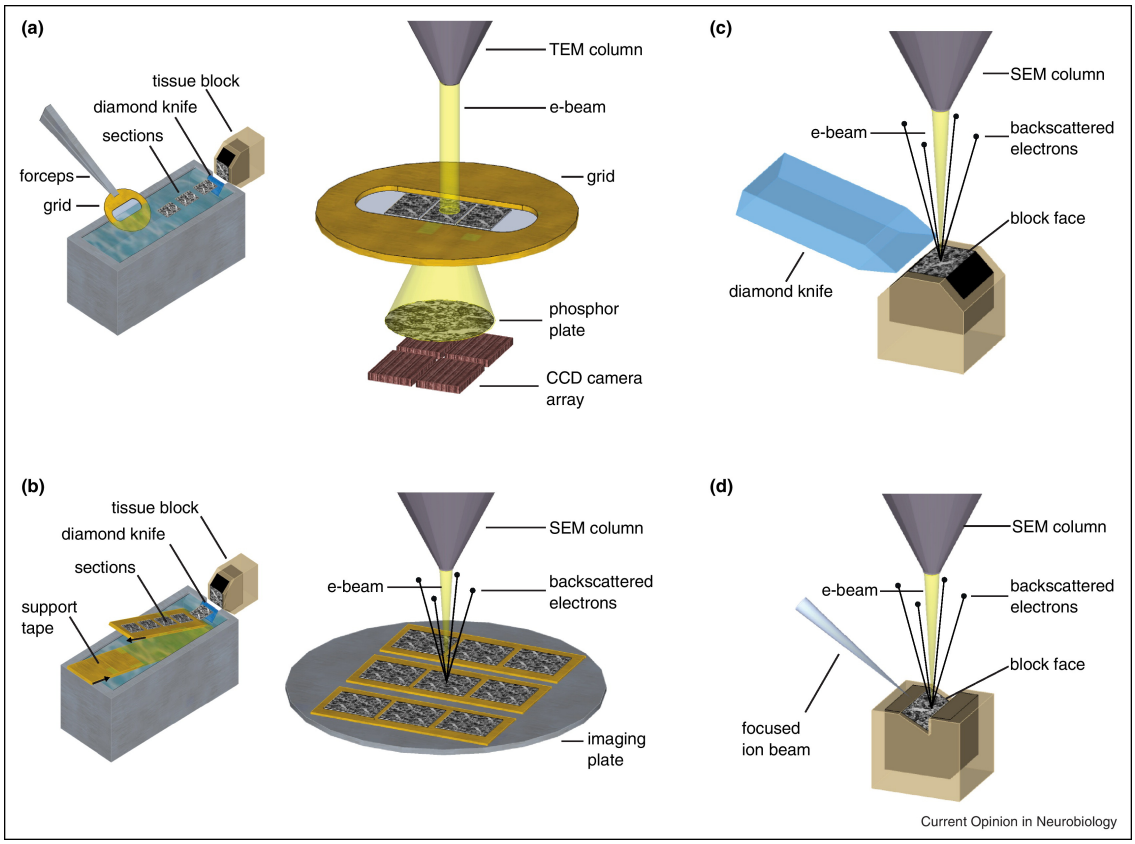

Figure 2.4: Overview of volume EM techniques. See subsection 2.1.2 for details. (a) Serial section transmission electron microscopy (ssTEM). (b) Automated tape-collecting ultramicrotome scanning electron microscope (ATUM-SEM). (c) Serial block face-scanning electron microscopy (SBEM). (d) Focused ion beam milling scanning electron microscopy (FIB-SEM). Reprinted with permission from "Volume electron microscopy for neuronal circuit reconstruction" by Kevin L. Brigmann and Davi D. Bock, 2012, Current Opinion in Neurobiology, 22, p. 156. Copyright 2011 by Elsevier Ltd.

subsequently combined into a three dimensional image stack. Such techniques are often referred to as 3D or volume EM (see Briggman and Bock 2012 for a review, and Figure 2.4). In brief, these methods provide different image resolution, slice thickness and maximal dataset dimensions. Two main approaches are used: Slice-based methods, which first cut the tissue sample into ultra-thin sections that are subsequently imaged by EM, and block-face methods, which alternate between imaging the surface of a block of tissue and abrading the topmost layer of the block.

In serial section transmission electron microscopy (ssTEM), ultra-thin slices of brain tissue are first cut and subsequently imaged using a regular transmission electron microscope. It is the oldest volume EM technique that was already used in the beginning of structural study of brain tissue using EM (Birch-Andersen, 1955) as well as for the first dense connectome of the whole nervous system of the nematode *C. elegans* (White et al., 1986). SsTEM allows a very high resolution of the imaged slices of typically only a few nanometers voxel size (in-plane resolution) with a slice thickness of about $50\,\mathrm{nm}$ (Harris et al., 2006). However, the manual handling of slices is error prone and can lead, for example, to folded slices or the loss of complete sections making it likely that processes cannot be traced anymore across a lost section. Section handling is automated in the automated serial section tape-collection scanning electron microscopy (ATUM-SEM) approach, which cuts the slices and collects them on a tape for later imaging (Hayworth

et al., 2006), allowing for a smaller slice thickness of typically $30\,\mathrm{nm}$

Block-face techniques make use of scanning electron microscopy (SEM) to image the surface of a block of tissue. Subsequently, the previously imaged surface is removed and the process is repeated. Hence, block face techniques do not suffer from section loss or image distortions thus facilitating data analysis. On the other hand, since the removal of the block face is a destructive process, it is not possible to image a slice again once it has been removed and it is not possible to parallelize imaging over different microscopes (Wanner et al., 2015). In focused ion beam scanning electron microscopy (FIBSEM), the block face is removed using an ion-beam to evaporate the tissue (Knott et al., 2008). FIBSEM allows for a very small cutting thickness of about $5\,\mathrm{nm}$ combined with a high in-plane resolution of a few nanometers as well. Thus, FIBSEM is able to generate high-resolution isotropic dataset making it the method of choice for circuits with small processes. However, the dataset size is currently limited to roughly $50\,\mu\mathrm{m}$ along the direction of the ion beam, although newer techniques are being developed to overcome this limitation (Hayworth et al., 2015). In serial block-face electron microscopy (SBEM), a diamond knife is used to cut away ultra-thin section from the block face (Denk and Horstmann, 2004). The slice thickness is limited to about $20\,\mathrm{nm}$ while the imaging resolution is typically about $10\,\mathrm{nm}$

The volume EM techniques presented above are currently largely complementary in terms of resolution and total dataset size (Lichtman and Denk, 2011). Except for FIBSEM, all volume EM techniques are in principle able to provide datasets with an edge length in the order of millimeters. Advances in EM imaging techniques like multi-beam scanning electron microscopy (mSEM) promise further speed up of close to two orders of magnitude in imaging time that will make larger or even whole brain datasets for some species possible with imaging times in the order of weeks rather than years (Eberle et al., 2015).

To image biological tissue using EM, specific sample preparation techniques are required that may affect tissue properties and the final EM images. Common EM staining methods cause tissue shrinkage, which alters the volume of processes, and the loss of extracellular space (ECS) that separates them (Korogod et al., 2015). More recently proposed sample preparation methods allow to preserve or even enhance ECS (Pallotto et al., 2015).

2.1.3. The Connectome

A comprehensive map of all connections in the brain is called a connectome or wiring diagram (Sporns et al., 2005). From a mathematical perspective, the connectome constitutes a graph consisting of neural elements and their connections. A graph is represented by an ordered pair $G = (V, E)$ of a set of vertices or nodes V and edges $E \subset V \times V$. If the edges have a direction, i.e. each edge $e \in E$ is an ordered tuple pointing from one node to another, the graph is called directed and undirected otherwise. In nanoscale connectomics, the nodes of the wiring diagram are given by single neurons and the edges correspond to synaptically coupled neurons that are directed for chemical synapses. Here, only chemical synapses are considered such that the connectome will be defined as a directed graph. The connectome is often represented by its

adjacency matrix $A = (a_{ij})_{i,j=1}^{N}$ whose entries a_{ij} are 1 if a directed connection between neuron i (the presynaptic or source neuron) and neuron j (the target or postsynaptic neuron) exists and 0 otherwise, where each neuron was assigned an integer label $1, \ldots, N$, i.e.

$$a_{ij} = \begin{cases} 1 & \text{if there is a synapse from neuron } i \text{ to neuron } j \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

A connectome specifying whether two neurons are connected or not is called a binary connectome. Since neurons can be connected by multiple synapses, the connectome can also be represented by a weighted adjacency matrix whose entries do not only contain the information that a connection exists but how many synapses it consists of, i.e. each entry a_{ij} is an integer specifying the number of synapses between the corresponding neurons. Note that these definitions are not universal, e.g. a weighted connectome could also use other correlates of the connection strength between two neurons such as total active zone area or the connectome could be defined as a multigraph where each synapse constitutes a single edge. The nodes and edges of the connectome are typically enriched with additional (biological) information, e.g. the neuron type for the nodes and the total contact area, the active zone/PSD area and synapse type or sign (excitatory, inhibitory) for the edges.

2.2. Computer Vision for Connectomics

Connectomics produces large amounts of data in the form of 3D digital images. The reconstruction of neural circuits from the images in an automated fashion requires analyzing their contents with computers which is the subject of the field of computer vision.

2.2.1. Digital Images

For the purposes of this thesis, a d-dimensional digital signal I is given by a mapping that assigns an intensity value to each signal location $x \in \mathcal{G} \subset \mathbb{Z}^d$, i.e.

$$I : \mathcal{G} \to \mathcal{I}, \tag{2.2}$$

where $\mathcal{I}$ is the set of possible intensity values and $d \in \mathbb{N}$ is the dimensionality of the signal. Digital images are 2D signals and their locations are called pixels while image stacks are 3D signals and their locations are called voxels. Signals are called binary if $\mathcal{I} = \{0, 1\}$, grayscale if $\mathcal{I} = \{0, \ldots, 255\}$ and RGB if $\mathcal{I} = \{0, \ldots, 255\}^3$. If the dimensionality of $\mathcal{I}$ is larger than one, e.g. corresponding to multiple measurements at each signal location, the signal is sometimes called a multi-channel signal. EM datasets are typically 3D grayscale image stacks. In practice, digital signals have a finite number of signal locations $|\mathcal{G}| < \infty$, which are typically arranged on a rectangular grid. For $d = 2$ and $\mathcal{G} = \{1, \ldots, N\} \times \{1, \ldots, M\}$ images can be conveniently represented as matrices $I \in \mathcal{I}^{N \times M}$, where N is the image height and M is the image width in pixels, and analogously for higher dimensions. For notational simplicity, signals are assumed to be zero outside their domain $\mathcal{G}$. The connectivity or neighborhood $\mathcal{N}(x) \subset \mathcal{G}$ of a signal location

$x \in \mathcal{G}$ is a set of locations around x, which are called the neighbors of x. For example, all image locations with a maximum distance (Chebyshev distance) less than one to the pixel of interest result in the 8 neighborhood in 2D

$$\mathcal{N}_8(x) = \{y \in \mathbb{Z}^2 \mid \|x - y\|_{\max} \leq 1\} \tag{2.3}$$

and the 26 neighborhood in 3D images

$$\mathcal{N}_{26}(x) = \{y \in \mathbb{Z}^3 \mid \|x - y\|_{\max} \leq 1\}. \tag{2.4}$$

2.2.2. Image Transformations

The abstract definition of signals allows transforming them using mathematical operations with the goal of extracting signal features that provide a high-level understanding, e.g. the recognition of patterns and objects. For notational simplicity and clarity, this section introduces basic signal transformations for images, which analogously can be defined for general d-dimensional signals.

Geometrical transformations change the coordinate system of the location grid $\mathcal{G}$ resulting e.g. in a translation or rotation of an image. For practical purposes, the image locations are typically embedded in $\mathbb{R}^2$ and non-integer image locations are calculated by interpolation from nearby locations, which for example can be necessary for rotations by arbitrary angles or non-rigid image deformations.

Image filtering operations apply transformations to the intensity values of an image. In its most general form, a filter f takes an input image $I : \mathcal{G} \to \mathcal{I}$ and produces an output image $fI : \mathcal{G} \to \tilde{\mathcal{I}}$, where $\mathcal{I}$ and $\tilde{\mathcal{I}}$ are the intensity values of the image and the filter response respectively, which can be different from each other. Thus, a filter constitutes an operator

$$f : \mathcal{G}^{\mathcal{I}} \to \mathcal{G}^{\tilde{\mathcal{I}}}, \tag{2.5}$$

where $\mathcal{G}^{\mathcal{I}} := \{f : \mathcal{G} \to \mathcal{I}\}$ denotes the set of all functions from $\mathcal{G}$ to $\mathcal{I}$. Important examples of image filtering operations for binary images are morphological operations, which allow the manipulation of the shape of objects. Two fundamental morphological operations are the erosion and dilation operation for expanding or contracting objects. The derived operations of opening and closing are used to remove small objects or close small holes.

An important part of many image filtering operations are linear filters, where the output is a linear function of the input pixel values. The discrete convolution is a frequently encountered example of a linear filter. For an image $I : \mathbb{Z}^2 \to \mathbb{R}$ and a filter kernel $w : \mathbb{Z}^2 \to \mathbb{R}$ the result of the discrete convolution $F : \mathbb{Z}^2 \to \mathbb{R}$ is given by

$$F(i,j) = (I * w)(i,j) := \sum_{k,l \in \mathbb{Z}} I(k,l)w(i-k, j-l) = \sum_{k,l \in \mathbb{Z}} I(i-k, j-l)w(k,l), \tag{2.6}$$

for $i, j \in \mathbb{Z}$, and analogously for higher dimensions. For a finite dimensional image $I \in \mathbb{R}^{N \times M}$ and a kernel $w \in \mathbb{R}^{k_1 \times k_2}$ with support in $\{-\lfloor k_1/2 \rfloor, \ldots, \lfloor k_1/2 \rfloor\} \times \{-\lfloor k_2/2 \rfloor, \ldots, \lfloor k_2/2 \rfloor\}$, this can also be written as

$$(I * w)(i,j) = \sum_{l_1=-\lfloor k_1/2 \rfloor}^{\lfloor k_1/2 \rfloor} \sum_{l_2=-\lfloor k_2/2 \rfloor}^{\lfloor k_2/2 \rfloor} w(l_1, l_2) I\left(r_1 i + l_1 d_1, r_2 j + l_2 d_2\right), \tag{2.7}$$

where $r = (r_1, r_2) \in \mathbb{N}^2$ is an optional stride and $d = (d_1, d_2) \in \mathbb{N}^2$ is an optional dilation ratio. A stride that is larger than one in dimension i corresponds to a subsampling of the output in this dimension by considering only every r_i-th pixel. For finite dimensional images, a strided convolution reduces the image size in dimension i by factor r_i. A dilation ratio that is larger than one in dimension i is equal to the convolution with a kernel where $d_i - 1$ zeros are inserted between all kernel entries in the corresponding dimension. For finite dimensional images and kernels, the output of the convolutional operation (Equation 2.7) can only be evaluated at locations that only depend on pixels in the input image and kernel domain, resulting in an output image with size $(N - (k_1 - 1)) \times (M - (k_2 - 1))$, which is sometimes called the valid convolution. The specification of boundary conditions such as a zero-padding of the input can result in larger outputs that can have the same size as the input image ("same convolution") and a maximal output size of $(N + k_1 - 1) \times (M + k_2 - 1)$ ("full convolution"). The convolution is a linear and associative operation. An important property of the convolution is translation equivariance, i.e. if the image pixels are shifted, then the output of a convolution is shifted by the same translation. More formally, let $x \in \mathbb{Z}^d$ and T_x be the translation operator acting on a d-dimensional signal f as $T_x f(y) = f(x + y)$, i.e. it is a geometrical transformation that shifts the signal locations, then

$$T_x(f * g) = (T_x f) * g = f * (T_x g). \tag{2.8}$$

Thus, a convolutional filter detects the same features in terms of the response magnitude in different parts of the image. A convolutional kernel w is called separable if it can be written as a convolution of other kernels w_i

$$w = w_1 * w_2 * \cdots * w_n. \tag{2.9}$$

The separability of a kernel can be used to reduce the computational complexity of a filter. Assume that a 2D kernel can be written as the convolution of two 1D kernels $w = w_1 * w_2$, where 1D refers to a kernel that only has non-zero entries along one axis. Then, due to the associativity of the convolution, a convolution of an image $I \in \mathbb{R}^{N \times M}$ with this kernel can be calculated as

$$I * w = I * (w_1 * w_2) = (I * w_1) * w_2 \tag{2.10}$$

resulting in two 1D convolutions along different axes instead of one 2D convolution. However, in a naive implementation of the convolution (not using fast Fourier transform (FFT)), the 2D convolution requires NMk_1k_2 multiplications and $NM(k_1k_2 - 1)$ additions (same boundary condition), whereas the two consecutive 1D convolutions require $NM \sum_i k_i$ multiplications and $NM(\sum_i k_i - 1)$ additions, which scales more favorably with respect to the kernel size resulting in

a faster computation and less memory usage. The construction of image transformations that yield useful features for a task is called feature design. This can e.g. consist of the explicit specification of a convolutional kernel. An alternative to feature design is to learn relevant features from the data using machine learning as discussed in the next section.

2.2.3. Image Segmentation

Many connectomic reconstruction tasks can be formulated as segmentation tasks, i.e. the grouping of voxels into segments (sometimes also called supervoxels or regions) that typically represent the objects of interest or boundaries between objects (see also Pal and Pal, 1993; Pham et al., 2000). A segmentation S of a d-dimensional signal $I : \mathcal{G} \to \mathcal{I}$ is a map

$$S : \mathcal{G} \to \mathcal{L}, \tag{2.11}$$

where $\mathcal{L} \subset \mathbb{N}$ is a set of segmentation ids. A segment consists of all locations with the same id i, i.e. the preimage of $S^{-1}(i)$ of i, and is thus uniquely identified by its id. Segments are sometimes also called regions or supervoxels. Note that in contrast to classical image segmentation, segments are not required to be connected, which is sometimes called pixel classification (Pham et al., 2000). An example of a segmentation in connectomics is a volume segmentation of EM data, which groups all voxels belonging to the same cellular process. The segments in a volume segmentation are given by individual processes which are assigned arbitrary unique ids. Further examples are given by the segmentation of ultrastructure such as mitochondria. In this case, one possibility would be to assign binary labels $L = \{0, 1\}$ corresponding to mitochondria and background, which in turn could be turned into a segmentation of individual mitochondria by further processing the binary maps e.g. using connected components.

The region adjacency graph of a segmentation consists of the segments as nodes with edges between adjacent segments. The adjacency of segments is typically defined using a neighborhood relation of the underlying voxels. The volume segmentations of 3D EM data considered here are often generated using the watershed segmentation algorithm (Beucher and Lantuéjoul, 1979), which can produce a segmentation in which segments are separated by a one-voxel boundary such that voxels within a segment only have other voxels from that segment or wall voxels in their 26 neighborhood. Wall voxels are assigned the distinguished id 0. The region adjacency graph for such a volume segmentation consists solely of segments with a positive id, i.e. wall voxels are ignored, and two segments are called adjacent if there is a wall voxel that contains ids from both segments in its 26 neighborhood. Volume segmentations can contain two kinds of errors: Two segments that belong to the same biological process constitute a split error that is associated with the corresponding edge of the supervoxel graph. A segment that spans two or more biological processes is called a merge error which is associated with the corresponding segment. The association of segmentation errors with different objects of the region adjacency graph, namely its edges for split errors and its nodes for merge errors, highlights the conceptual difference of the errors. Split errors can be resolved by an agglomeration procedure, which removes edges from the supervoxel graph that constitute split errors and merges the corresponding segments. In terms of the underlying segmentation, this only requires a relabeling of segmentation ids, which can

even be done virtually by introducing equivalence classes of segments from the same biological process, i.e. lists of segments in the same processes. Merge errors on the other hand require to split a segment into several parts, which typically requires to either correct the segmentation locally or to allow segments to be part of multiple objects, e.g. using overlapping segments lists. A segmentation is called an oversegmentation if it has the tendency to split the target objects into multiple segments and an undersegmentation if it has the tendency to merge objects.

2.3. Machine Learning

Machine learning aims to provide algorithms that can automatically analyze data without being explicitly programmed (Samuel, 1959). Instead of following static instructions to solve a particular task, the relevant decision criteria are learned from sample data. The data-driven approach of machine learning enables the detection of domain-specific patterns that can be used to make predictions for unseen samples or solve decision making tasks. This chapter is based on Murphy (2012), Bishop (2007) and Goodfellow et al. (2016).

2.3.1. Supervised Learning

The goal of supervised learning is to infer a relation between inputs $x \in X$ and outputs $y \in Y$ given a set of labeled training samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where N is the number of training samples and X and Y are the input and output space, respectively. Labeled in this case means that for each input x_i the corresponding output y_i is known. In the simplest case, the relation between inputs and outputs is given by a function $f : X \to Y$. More generally, a probabilistic approach specifies the relation by a conditional distribution $p(y|x)$. If the data model $p(y|x)$ is specified using a fixed set of parameters θ, the model is called a parametric model, otherwise, if the number of parameters grows based on the training data, the model is called nonparametric. The input and output spaces X and Y can in principle be any set. General inputs are typically represented in a feature space by a mapping $\phi : X \to \mathbb{R}^d$ that describe characteristic properties of the object. For example, in image processing the raw input data are typically images of a fixed size $N \times M$ resulting in the input space $X = \mathbb{R}^{M \times N}$ and a feature representation of an image could be the result of a set of image filters called a feature map. Choosing a good feature representation, a process known as feature construction or design, is an important factor for many machine learning models and is often done by human experts. Selecting the most predictive input features for a task from a large set of features is known as feature selection. Algorithms such as artificial neural networks are able to automatically learn hierarchical feature representations directly from the input data (see subsection 2.3.4). If Y is a finite categorical set, i.e. without loss of generality (wlog) $Y = \{1, \ldots, C\}$, then the problem is called a classification or pattern recognition task while for $Y = \mathbb{R}^d$ it is called a regression task.

For parametric models the learning or training consists of estimating the parameters given the training data set. In the non-probabilistic setting, the parameters can be fitted by minimizing an error or loss function $L(\eta(x), y)$ on the training data, where η is the model prediction for an input x and y is the corresponding true output value. There is, however, no distinguished loss function

in the non-probabilistic setting. A simple choice for a classification task could be the 0-1 loss $L(\eta, y) = \mathbb{I}(\eta = y)$, where $\mathbb{I}$ is the indicator function given by

$$\mathbb{I}(\eta = y) = \begin{cases} 1 & \text{if } \eta = y \\ 0 & \text{otherwise.} \end{cases} \tag{2.12}$$

However, the 0-1 loss is not smooth or even differentiable, which is why surrogate loss functions are often used. Examples are the hinge loss $L(\eta, y) = \max(0, 1 - \eta y)$, which is used for classification in support vector machines (SVM), or the exponential loss $L(\eta, y) = \exp(-\hat{y}_i \eta)$ for AdaBoost (see 2.3.3), where $\hat{y}_i \in \{-1, 1\}$. In the probabilistic setting, there are more natural choices for parameter estimators. The full knowledge about the parameters is given by the posterior distribution $p(\theta|\mathcal{D})$, which in most cases is analytically intractable or computationally difficult to calculate. Instead, point estimators are often considered that result in some frequently used loss function. A common estimator is the maximum likelihood estimate (MLE)

$$\hat{\theta} = \arg\max_{\theta} \log p(\mathcal{D} \mid \theta). \tag{2.13}$$

The MLE is equivalent to the minimization of the negative log likelihood (NLL)

$$\text{NLL}(\theta) = -\log p(\mathcal{D} \mid \theta). \tag{2.14}$$

If the training samples are assumed to be independent and identically distributed, the NLL can be written as

$$\text{NLL}(\theta) = -\sum_{i=1}^{N} \log p(y_i \mid x_i, \theta). \tag{2.15}$$

For a Gaussian model of the form $p(y \mid x, \theta) = \mathcal{N}(y \mid f_\theta(x), \sigma^2)$, where $\mathcal{N}$ is the density function of the Gaussian distribution, $f_\theta : X \to Y$ is a function with parameters θ and σ is the fixed standard deviation, the minimization of the NLL is equivalent to the minimization of the sum of squares error

$$L_{\text{se}}(\theta) = \frac{1}{2} \sum_{i=1}^{N} (y_i - f_\theta(x_i))^2, \tag{2.16}$$

which is often used for regression tasks. Assuming a fixed but different standard deviation for each training sample results in the weighted sum of squares error

$$L_{\text{se}}(\theta) = \frac{1}{2} \sum_{i=1}^{N} w_i (y_i - f_\theta(x_i))^2, \tag{2.17}$$

where $w_i \in \mathbb{R}$ are the weights associated with each training sample. Similarly, for a binary model of the form $p(y \mid x, \theta) = \mathcal{B}(y \mid \sigma(f_\theta(x)))$, where $\mathcal{B}$ is the Bernoulli distribution, $\sigma(x) = \frac{1}{1+\exp(-x)}$ and $f_\theta : X \to Y$, the NLL results in the cross-entropy error

$$\text{NLL}(\theta) = -\sum_{i=1}^{N} y_i \log \sigma(f_\theta(x_i)) + (1 - y_i) \log(1 - \sigma(f_\theta(x_i))), \tag{2.18}$$

which is often used for binary classification. Using the softmax function for $C \in \mathbb{N}$ classes

$$\sigma : \mathbb{R}^C \to [0,1]^C, \sigma_i(x) = \frac{\exp(x_i)}{\sum_{j=1}^{C} \exp(x_j)}, \tag{2.19}$$

where σ_i denotes the i-th component of σ, this can be generalized to a multi-class classification $y \in \{1, \ldots, C\}$ using the model $p(y \mid x, \theta) = \sigma(f_\theta(x))$ with $f_\theta : X \to \mathbb{R}^C$ resulting in the cross-entropy error for multiple classes

$$\text{NLL}(\theta) = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log \sigma(f_\theta(x_i))_c. \tag{2.20}$$

Another common parameter estimator is the maximum a posteriori (MAP)

$$\hat{\theta} = \arg\max_\theta \log(p(\mathcal{D} \mid \theta)p(\theta)) = \arg\max_\theta \log p(\mathcal{D} \mid \theta) + \log p(\theta), \tag{2.21}$$

where $p(\theta)$ is the prior distribution over the model parameters. Assuming $p(\theta) = \mathcal{N}(0,1)$ this results in the L2-regularization term

$$\|\theta\|_2^2 = \sum_k \theta_k^2 \tag{2.22}$$

that is added to the MLE estimate. In the following, two particular machine learning models and examples of learning algorithms to train them are introduced.

2.3.2. Decision Trees

Decision trees are a popular machine learning model due to their simplicity and interpretability (see also Murphy, 2012). The input space $X \subset \mathbb{R}^d$ is recursively partitioned by axis aligned splits using each partition as the input for the next step. The final model can be written as

$$f(x) = \sum_i w_i \mathbb{I}(x \in R_i), \tag{2.23}$$

where $R_i \subset X$ is the i-th region and $w_i \in Y$ is the region prediction, i.e. a real value for regression or ordinal values for classification tasks. The evaluation of a decision tree for an input x consists of traversing a binary tree until a leaf node containing the tree response is reached. The child node of a node k is selected by a simple thresholding procedure $x_{j_k} \leq t_k$, where $t_k \subset \mathbb{R}$ is the threshold at node k of the tree and x_{j_k} is the j_k-th dimension of x. The learning consists of determining the structure of the tree as well as the decision threshold t_k and feature dimension j_k for each node k.

Since finding the optimal partition is NP-complete (Hyafil and Rivest, 1976), decision trees are typically constructed in a greedy fashion. This is for example done in the classification and regression trees (CART) framework that computes a locally optimal MLE solution (Breiman et al., 1984). Consider a node of a binary decision tree and a set of training data points $\{x_i, y_i\}$. The decision criterion for the node is determined by choosing a feature j and a threshold t that results

in the smallest loss at this node, i.e.

$$(\hat{j}, \hat{t}) = \arg \min_{j=\{1,\dots,d\}} \min_{t \in T_j} L(\{x_i, y_i : x_{ij} \leq t\}) + L(\{x_i, y_i : x_{ij} > t\}), \tag{2.24}$$

where T_j is the set of possible thresholds for dimension j, e.g. the unique values of all training samples for the corresponding dimension and x_{ij} denotes the j-th dimension of the data point x_i. Typical loss functions are the squared loss with respect to the mean of the data for regression and the misclassification rate, entropy or gini index for classification. If the optimal feature j and threshold t are determined, the training data is distributed accordingly and the procedure is repeated for both newly constructed nodes. The splitting stops if predefined criteria are met, e.g. if the tree reaches a maximal depth or the data in the node is sufficiently homogeneous (e.g. all labels are in the same class). If a node is not split anymore, its response is calculated from the data in the node either as the mean target value of or from the class label distribution.

2.3.3. Ensemble Methods and Boosting

Ensemble methods combine weighted base models into a larger model of the form

$$F_M(x) = \sum_{m=1}^{M} w_m f_m(x), \tag{2.25}$$

where w_m is the weight for the m-th base model f_m. One approach to construct ensembles is the meta-learning technique boosting that typically uses an iterative training procedure to construct and combine base models called weak learners. Probably the most famous boosting algorithm for binary classification is AdaBoostM1 (Freund and Schapire, 1997) that optimizes the exponential loss $L(F_M(x), y) = \exp(-yF_M(x))$, where $y \in \{-1, 1\}$. In each iteration of the AdaBoostM1 training algorithm, the classifiers F_{n-1} of the previous steps are fixed and the current classifier f_n is trained to minimize

$$L(f_n) = \sum_i \exp\left(-y_i(F_{n-1}(x_i) + \beta f_n(x_i))\right), \tag{2.26}$$

where $\beta \in \mathbb{R}$ is a constant that is optimized in addition to f_n. The solution to the optimization problem is a classifier f_n trained on a weighted version of the training data set where the weights are determined by the errors of the previous models F_{n-1} (see Freund and Schapire, 1997 and Murphy, 2012 for details). However, due to the use of the exponential loss, AdaBoostM1 is sensitive to outliers and does not admit a probabilistic interpretation. To overcome these issues, boosting algorithms have been developed for other loss functions, e.g. the LogitBoost algorithm (Friedman et al., 2000) that uses the log-loss

$$L(F_M(x), y) = \log(1 + \exp(-yF_M(x))), \tag{2.27}$$

where $y \in \{-1, 1\}$ as before. By its very definition the log-loss is the logarithm of a probability mass function and outliers only contribute linearly.

2.3.4. Artificial Neural Networks and Deep Learning

Artificial neural networks (ANNs) are inspired by functional principles of biological neural networks. In the following, feed-forward ANNs are considered as they are typically used in deep learning algorithms for computer vision. ANNs consist of artificial neurons that are wired together with learned connection. Each artificial neuron calculates a weighted sum of its inputs $x \in \mathbb{R}^d$ that is passed through a non-linear activation function h resulting in the response $z \in \mathbb{R}$ of the neuron given by

$$z = h\left(w_0 + \sum_{i=1}^{d} w_i x_i\right) = h\left(w_0 + w^T x\right), \tag{2.28}$$

where $w_i \in \mathbb{R}$ are input weights of the neuron and $w^T x$ denotes the scalar product of vectors. For notational simplicity the bias term w_0 can be subsumed into the input weights by extending the input vector with an additional dimension with fixed value one, i.e. by defining $\hat{x} = (1, x) \in \mathbb{R}^{d+1}$. The non-linearity h is typically chosen as a differentiable sigmoidal function such as the logistic sigmoid, or the hyperbolic tangent. Other non-linearities such as the rectified linear unit (relu) given by

$$\text{relu}(x) = \max(0, x), \tag{2.29}$$

which is differentiable almost everywhere, have been found to be superior in many applications (Glorot et al., 2011). In fully-connected feed forward networks, the neurons are arranged in layers with each neuron receiving input from all neurons in the previous layer. The output $z_l \in \mathbb{R}^N$ of layer l consisting of N neurons can be conveniently expressed as a matrix multiplication of a weight matrix W_l where each row corresponds to the weights of one neuron in the layer with the output of the previous layer z_{l-1} followed the point-wise application of a non-linearity h

$$z_l = h(W_l z_{l-1}). \tag{2.30}$$

The neuron activities in the last layer L contain the network predictions $f(x)$ and the layer is called the output layer of the network. The layers between the input layer and the output layer are called hidden layers. An ANN with a single hidden layer is a universal approximator, i.e. it can model suitably smooth functions arbitrarily close given enough hidden units (Hornik, 1991). However, deeper networks consisting of several hidden layers are often computationally more efficient (in terms of parameters) when representing a function compared to shallow networks with only a few hidden layers (Bengio et al., 2009). The term deep learning refers to the usage of ANNs models with more than one hidden layer.

A standard way for training ANNs is to assume that the output constitutes the mean of a Gaussian model $p(y|x) = \mathcal{N}(y|f(x), \sigma^2)$ with fixed standard deviation σ or a binomial model $p(y|x) = \mathcal{B}(y|f(x))$. The MLE for these models corresponds to the minimization of the sum of squares error (Equation 2.16) or the cross-entropy error (Equation 2.18), respectively. If the non-linearities in the network are differentiable almost everywhere, the minimization of the loss function $L = L(\theta_t)$ can be done by gradient descent, i.e. by iteratively changing the model parameters θ in the opposite direction of the gradient of the loss function $\nabla_\theta L(\theta_t) = \frac{\partial L(\theta_t)}{\partial \theta}$. The calculation of the

error gradient involves the application of the chain rule to calculate the gradients of the parameters in the hidden layers and is called the backpropagation algorithm (Rumelhart et al., 1986). In the simplest version of gradient descent, the parameters θ are updated by a scaled version of the gradient

$$\theta_{t+1} \to \theta_t - \eta_t \nabla_\theta L(\theta_t), \tag{2.31}$$

where $\eta_t > 0$ is the learning rate that constitutes an optimization hyperparameter that can depend on the current iteration. Note that the sum of squares error and the cross-entropy error can be written as a sum of terms for each training examples that can be interpreted as an expectation with respect to the data distribution $p_\mathcal{D}$ on $X \times Y$

$$L(\mathcal{D}) = \sum_{i=1}^{N} L(f(x_i), y_i) = \mathbb{E}_{x,y\sim p_\mathcal{D}}[L(f(x), y)]. \tag{2.32}$$

This expectation can be approximated by sampling only a small number of training examples from the training data called a minibatch. Training the network with gradients calculated from minibatches is called batch gradient descent. In the extreme case of a single training example it is called stochastic gradient descent. Batch and stochastic gradient descent result in a noisy estimate of the gradient which, however, is much faster to compute in particular for very large datasets and also allows for online training. A simple but typically substantial acceleration method of the default gradient descent algorithm is to update the velocity instead of the position in weight space by adding a momentum term

$$\theta_{t+1} \to \theta_t + \Delta\theta_t = \theta_t + m\Delta\theta_{t-1} - \eta_t \nabla_\theta L(\theta_t), \tag{2.33}$$

where $\Delta\theta_t$ and $\Delta\theta_{t-1}$ are the current and previous weight update, respectively, and $m \in (0, 1)$ specifies the relative contribution of the momentum term to the current gradient (Rumelhart et al., 1986). Note that other extensions of the default gradient descent method exist using e.g. adaptive learning rates such as in AdaGrad (Duchi et al., 2011) or Adam (Kingma and Ba, 2014).

2.3.5. Convolutional Neural Networks

Instead of using ANNs with fully connected layers, the network architecture can be constrained to include a-priori knowledge about the task. For structured data such as images, an efficient strategy is to maintain their structural properties such as the locality of pixels in the network architecture. Convolutional neural networks (CNNs) for image analysis achieve this by ordering the neurons in each layer in two-dimensional grids, where all neurons in one grid share the same parameters and receive input from local patches in the grids of the previous layer that are slightly shifted for each neuron (LeCun et al., 1989). The parameter sharing in CNNs results in a substantial reduction of trainable parameters and, together with the localized field of view, introduces translation equivariance of the calculated features. Consequently, the basic operation of CNNs is the convolution defined in Equation 2.6. For image data, a CNN typically operates on 3D image stacks $z \in \mathbb{R}^{N_1 \times N_2 \times c}$, where N_1, N_2 are the spatial dimensions of the data, i.e. width and height for images, and c is the number of channels also called number of feature maps. If the network is trained on raw image data then the channels of the input layer are set corresponding to the type

of input image, i.e. $c_1 = 1$ for grayscale images or $c_1 = 3$ for RGB images. Each layer l of a CNN in its most basic form calculates multiple 3D convolution of the output of the previous layer $z^{l-1} \in \mathbb{R}^{N_1^{l-1} \times N_2^{l-1} \times c_{l-1}}$ with kernels $w_i^l \in \mathbb{R}^{k_{i1}^l \times k_{i1}^l \times c_{l-1}}$, $i = 1, \ldots, c_l$ to produce the outputs

$$z_i^l = h(z^{l-1} * w_i^l + b_i^l) \in \mathbb{R}^{N_1^l \times N_2^l \times 1}, \tag{2.34}$$

where $b_i^l \in \mathbb{R}$ is an additive bias. The single outputs z_i^l of a layer are stacked in the third dimension to again produce a 3D output $z^l \in \mathbb{R}^{N_1^l \times N_2^l \times c_l}$. Note that for the 3D convolution, the channel dimension is typically treated differently than the spatial dimensions, i.e. the size of the convolutional kernel in the channel dimension is equal to the total number of channels and there is typically no dilation or stride included in this dimension. This corresponds to the network doing separate 2D convolutions for each input channel that are added up to produce one output channel. The principle of convolutional neural networks is not limited to 2D images but can be generalized to arbitrary d-dimensional signals such as audio and image stacks/videos.

Another common operation in CNNs are pooling layers that combine feature map responses of nearby pixels and often reduce the size of the feature maps in a layer. The most prominent example is the max-pooling layer that corresponds to a strided maximum filter. In brief, a max-pooling layer calculates the maximal response in a rectangular neighborhood of all pixels on a regular grid of a feature map, i.e. in the case of 3D network layers

$$z_{l+1}(x, y, z) = \max_{w \in W, h \in H} z_l(xk_1 + w, yk_2 + h, z), \tag{2.35}$$

where $W, H \subset \mathbb{Z}$ define the size of the max-pooling window and $k_1, k_1 \in \mathbb{N}$ the pooling stride. The pooling is typically done separately for each image channel as defined in Equation 2.35. The size of the output in dimension i is smaller by a factor k_i. Other examples of pooling operations are average pooling that calculates the average of each input window or strided convolutions that only evaluate the output of the convolution on a sparse grid instead of all pixel locations.

Each layer of ANNs described above constitutes a differentiable transformation which are chained together to give the network result. The chain structure can be generalized to directed acyclic computation graphs representing a differentiable transformation of an input variable to an output variable. Since the composition of differentiable transformations is differentiable, the parameter gradients with respect to some loss function can be calculated by automatic differentiation (Rall, 1981) allowing for optimization by gradient descent. Thus CNNs are not limited to the strict sequence of layers as described above but can contain e.g. skip connections or parallel processing pathways. Furthermore, other transformations that are differentiable can be used in the computation graph as well such as the fully connected layers described in the previous section. However, a fully connected layer takes an input of a fixed size and produces a fixed-sized output whereas a convolutional or pooling layer can process inputs of arbitrary size. A CNN is called a fully convolutional network (FCN) if it can process inputs of arbitrary size producing spatially dense outputs of corresponding size (Long et al., 2015; Springenberg et al., 2014). The receptive field of an output pixel from a FCN is given by all pixels in the input image that contribute the output

value. Putting it another way, a default CNN can be thought of as non-linear function while a FCN is non-linear filter (Long et al., 2015).

2.3.6. Performance Evaluation Metrics

Ultimately, the goal of using machine learning models is to make predictions on data that has not been used during training. To estimate the performance on novel data, the generalization error of a model f is defined as the expected error with respect to some loss function L and the distribution $p(x, y)$ on $X \times Y$ that is used for future predictions (see also Murphy, 2012; Goodfellow et al., 2016), i.e.

$$\mathbb{E}_{p(x,y)}[L(f(x), y)]. \tag{2.36}$$

The distribution $p(x, y)$ for which the generalization error is calculated is typically the same distribution from which the training set was generated. However, the training error is typically not a good estimator for the generalization error because complex models might be able to perfectly fit the training data but show a high generalization error which is called overfitting. The generalization error can be approximated by an independent test set, also called ground truth, which was not used for training.

In general, the loss function L might not contain all information of interest, e.g. the 0-1 loss does not include whether a specific class is misclassified while the exponential loss of AdaBoost is difficult to interpret for a classification task, motivating the use of additional quantities to measure the algorithm performance. For binary classification, a set of common performance evaluation quantities are the confusion matrix, the receiver operator characteristic (ROC) curve, precision, recall, area under curve (AUC) and the F1 score (see also Murphy, 2012). Let $y \in \{0, 1\}$, where $y = 1$ is called the positive class and $y = 0$ the negative class, which is known for all data points in the test set. Furthermore, let $f(x)$ be the output of a classifier, e.g. the probability $p(y = 1|x)$ or another measure of confidence that $y = 1$. The classification decision amounts to choosing a threshold τ and assigning the predicted class $\hat{y}$ as

$$\hat{y}(x) = \begin{cases} 1 & \text{if } f(x) \geq \tau \\ 0 & \text{otherwise.} \end{cases} \tag{2.37}$$

For each choice of τ, the confusion matrix reports the classification performance by counting the number of true positive (TP: $y = \hat{y} = 1$), false positive (FP: $y = 0, \hat{y} = 1$), false negatives (FN: $y = 1, \hat{y} = 0$) and true negative (TN: $y = \hat{y} = 0$) predictions that occur in the test set (see Table 2.1). Typically, the confusion matrix is not used directly for performance assessment but error metrics derived from it (see Table 2.1). The recall or true positive rate specifies the fraction of retrieved positive ground truth instances

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{2.38}$$

		Ground truth $y = 1$	Ground truth $y = 0$	
Prediction	$\hat{y} = 1$	TP	FP	Precision: $\frac{TP}{TP+FP}$
	$\hat{y} = 0$	FN	TN	
		Recall: $\frac{TP}{TP+FN}$	FPR: $\frac{FP}{FP+TN}$	

Table 2.1: **Confusion matrix and derived error metrics.** TP: true positives, FP: false positives, FN: false negatives, TN: true negatives, recall (Equation 2.38), FPR: false positive rate (see Equation 2.39) and precision (Equation 2.40).

The false positive rate (FPR) is given by

$$\text{FPR} = \frac{FP}{TN + FP}. \tag{2.39}$$

The precision specifies the fraction of relevant correct instances among all positively classified instances

$$\text{Precision} = \frac{TP}{TP + FP}. \tag{2.40}$$

Instead of using a fixed threshold τ, the confusion matrix entries and the derived quantities can be considered as functions of τ. The ROC curve consisting of the FPR and the recall is often used to assess the performance of a classifier. However, for a large class-imbalance with many more negative than positive samples, the FPR is typically very small making the ROC curve difficult to read. Furthermore, in tasks such as object detection, there might not even be a reasonable way to define the negative class. In these cases, precision and recall are often used to describe the classifier performance because they only depend on the positive class. To compare ROC curves or precision recall curves from different classifiers, there are several ways to summarize the curves by a single number. A common way to do this is to calculate the AUC score that corresponds to the total area below the respective curve. The F1-score is an alternative performance metric that combines precision and recall into a single quantity by taking the harmonic mean

$$F_1 = 2\frac{1}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}. \tag{2.41}$$

3. Automated Synapse Detection for EM-based Connectomics

In this chapter, a novel approach for synapse detection in large-scale 3D electron microscopy (EM) datasets called SynEM is described. SynEM detects synapses by classifying interfaces between neuronal processes extracted from a volume segmentation of the data. In section 3.3, the SynEM approach to synapse detection is defined and evaluated in section 3.4 on two datasets imaged using serial block-face electron microscopy (SBEM) and automated serial section tape-collection scanning electron microscopy (ATUM-SEM). For applications in connectomics, the neuron-to-neuron errors are estimated based on the prediction errors of single synapses (subsection 3.3.4) showing that SynEM achieves precision and recall rates above 97% for binary cortical connectomes (subsection 3.4.5).

This chapter is an extension of the publications Staffler et al. (2017a,b).

3.1. Introduction

Synapse detection in large-scale 3D EM datasets, although considered an easier task than neurite reconstruction, is currently the bottleneck in connectomic data analysis and more than a factor of three slower than neurite reconstruction in terms of manual annotation time as outlined in section 1.2 (see also Dorkenwald et al., 2017; Staffler et al., 2017a). From a connectomics perspective, the foremost goal of synapse detection is to determine whether two neurons are connected. This requires the detection of the synapse itself, but also the identification of the corresponding pre- and postsynaptic neuron which is called the partner detection task. Conceptually, synapse detection requires the localization of a precise arrangement of contextual cues: A vesicle cloud agglomeration at the presynaptic membrane, a synaptic cleft and a postsynaptic density (PSD). In addition, further properties of synapses can be of interest such as the area of the active zone or the PSD, which can serve as structural correlates for synaptic strength (Bartol Jr et al., 2015). The task of partner detection requires one to discern the correct postsynaptic partner or partners among all incidental touches of the presynaptic bouton. If neuronal processes are represented by a volume segmentation, partner detection amounts to the specification of the pre- and postsynaptic segmentation id of a synapse. Thus, synapse detection for connectomics typically needs to take both the raw image data for the localization of synapses as well as the segmentation for partner identification into account.

The focus of this thesis is on synapse detection for cortical synapses obtained using SBEM. Due to the in-plane resolution of typically around $10\,\mathrm{nm}$, the appearance of synapses can be slightly different from high resolution data. Figure 3.1 shows an example of an excitatory spine synapse at a pixel size of $11.24 \times 11.24\,\mathrm{nm}^2$. Note that single vesicles might not be discernible, however, the shape of the vesicle cloud as a whole is still clearly outlined and in particular it is possible to

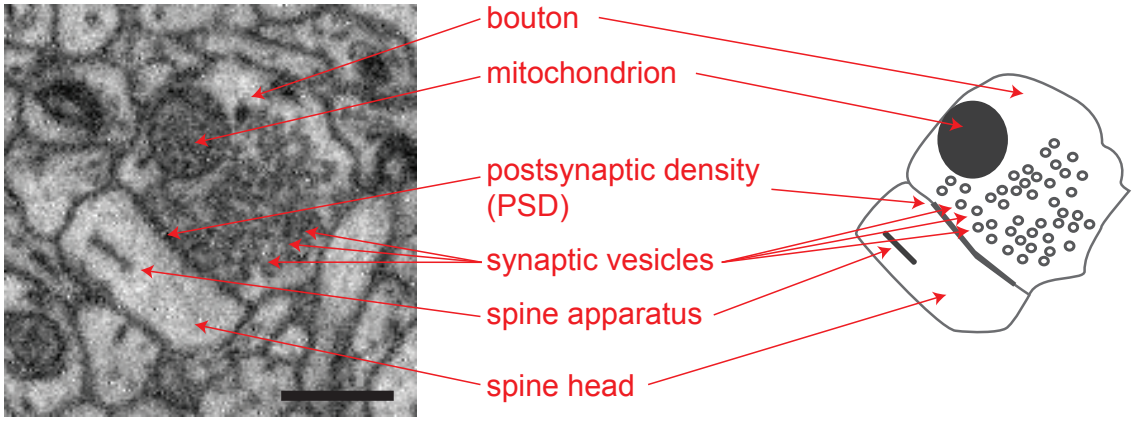

Figure 3.1: Excitatory synapse in serial block-face EM. Example of an excitatory synapse onto a spine head in a single slice at a pixel size of $11.24 \times 11.24\,\mathrm{nm}^2$ from the serial block-face EM dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b). The synaptic bouton is filled with vesicles that only attach to the membrane towards the postsynaptic spine head accompanied by a darkening and thickening of the corresponding membrane due to the postsynaptic density (PSD). Scale bar: 500 nm (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, on page 1 in `https://doi.org/10.7554/eLife.26414.034` licensed under CC BY 4.0)

see whether there is an actual apposition of the vesicle cloud to a membrane. Furthermore, there is typically no synaptic cleft visible between pre- and postsynaptic process and the PSD can be less pronounced than in data with higher resolution (cf. Figure 2.3), which can make the identification of the active zone and hence the postsynaptic partner detection more difficult in particular for inhibitory synapses (Figure 3.2). Although the identification of synapses from a single slice might be difficult, the volume imaging substantially improves synapse detection, because it is possible to inspect synapses in multiple consecutive slices (Helmstaedter, 2013; see also Supplementary Material 4 in Staffler et al., 2017a for a description of synapse detection in SBEM datasets). Due to the resolution limitation, the distinction of excitatory (Gray type I or asymmetric) and inhibitory (Gray type II or symmetric) synapses according to structural properties, such as the shape of vesicles and the cleft size, is often difficult even when using the 3D context. However, a further advantage of connectomic datasets is that several synapses along an axon can be inspected. Since inhibitory synapses are predominantly innervating dendritic shafts and somata (see e.g. Kubota et al., 2016; Santuy et al., 2018), the distribution of targets along axons can be used in addition to structural properties of single synapses.

Despite previous work on automated synapse detection approaches, methods for datasets with medium in-plane resolution as produced by SBEM are still scarce and the methods developed for data with high in-plane resolution such as focused ion beam scanning electron microscopy (FIBSEM) or serial section transmission electron microscopy (ssTEM) often show a considerable drop in performance when applied to medium resolution data.

Here, a novel method for automated synapse detection in 3D EM data called SynEM is presented. SynEM is based on an automated volume segmentation and combines synapse and partner detection into a single classification task. SynEM achieves 88% precision and recall rates for cortical synapses in a SBEM dataset and estimated binary neuron-to-neuron precision and recall rates of 97%.

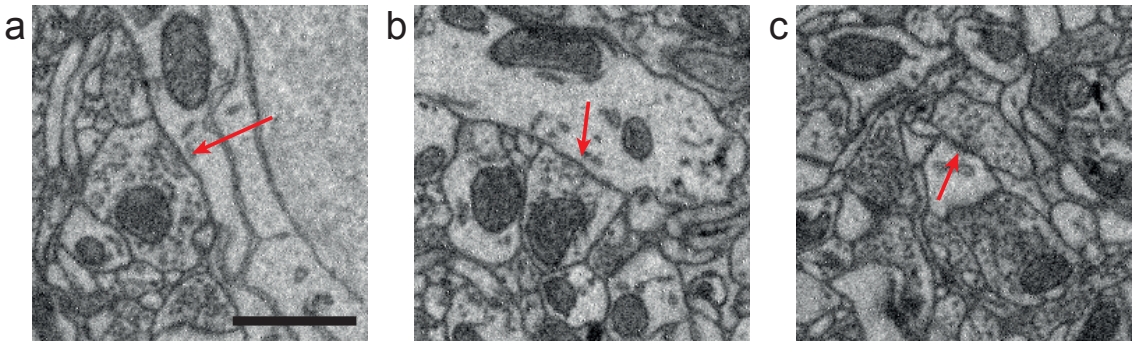

Figure 3.2: Tentatively inhibitory synapse in SBEM. All images are from a dataset from mouse primary somatosensory cortex (S1) layer 4 (L4) (2012-09-28_ex145_07x2; Boergens and Helmstaedter, 2012b) at a pixel size of $11.24 \times 11.24\,\mathrm{nm}^2$ (a) Soma synapse of a large bouton with a clear vesicle agglomeration towards the somatic membrane (red arrow) but not towards any other process touching the bouton. (b) Shaft synapse of a medium-size bouton again indicated by the clear vesicle agglomeration only towards the shaft membrane (red arrow). (c) Secondary spine head innervation indicated by the vesicle agglomeration towards the common membrane (red arrow) and potentially a slight thickening of the postsynaptic membrane. Note the second more distinct synapse onto the spine head below the labeled synapse. Scale bar: 1 µm (a - c)

3.2. Related Work

Several methods for automated synapse detection have been proposed for different EM modalities (see also Supplementary File 1 in Staffler et al., 2017a). Mishchenko et al. (2010) used a 2D PSD recognition algorithm for ssTEM data that classifies voxels on boundaries between axonal and dendritic processes from a volume segmentation using features (integrals) calculated in the direction orthogonal to the membrane. The actual synapse detection and partner identification were combined by considering the total PSD score of all voxels on boundaries between axon-dendrite pairs. Kreshuk et al. (2011) used a voxelwise PSD classification based on a set of image filters for isotropic FIBSEM data and extended the approach in Kreshuk et al. (2014) by a subsequent object classification step for highly anisotropic ssTEM data. Becker et al. (2013) extended the method of Kreshuk et al. (2011) with contextual cues by considering additional locations around the voxel of interest that are placed in a local reference frame with a consistent orientation at synapse locations. Neila et al. (2016) did a voxelwise classification similar to Kreshuk et al. (2011), which was refined by a subsequent regularization using conditional random fields to improve the resulting segmentation, which they apply to mitochondria and synaptic junction segmentation. Perez et al. (2014) used a cascaded hierarchical model for pixelwise classification and binarization using active contours to segment different kind of ultrastructure (mitochondria, lysosomes, nuclei and nulceolus). The method was also considered for the task of synapse detection in Dorkenwald et al. (2017). Other approaches leverage the remarkable progress that has been made in computer vision in the last couple of years using deep learning techniques. Roncal et al. (2015) compared a voxelwise approach using image filters and a random forest classifier (vesicle-rf) to a 2D convolutional neural network (CNN) (vesicle-cnn) and showed that even using only 2D information the CNN performance is superior. While the previous approaches focus on voxelwise classification of a single type of ultrastructure such as PSD or mitochondria, the SyConn reconstruction approach (Dorkenwald et al., 2017) consisted of a multi-class voxelwise classification into mitochondria, vesicles and synaptic junctions using recursive 3D CNNs thus sharing the learned features and exploiting the co-occurrence of these classes. Heinrich et al. (2018) used a 3D encoder-decoder

type CNN for voxelwise prediction of synaptic clefts formulated as a regression task. Parag et al. (2018) used a 3D encoder-decoder type CNN to learn regression labels for pre- and postsynaptic partners, which were called signed proximities. In contrast to default synaptic cleft labels, signed proximities allow to infer the synapse orientation directly from the voxel predictions. Other approaches have been developed for synapse specific staining (Navlakha et al., 2013) or for synapses with features that are tissue-specific such as rabbit retinal ribbon synapses (Jagadeesh et al., 2014) or synapses in Drosophila (Plaza et al., 2014; Huang and Plaza, 2014).

The task of partner detection on the other hand has initially received less attention than synapse detection. Kreshuk et al. (2015) proposed an approach for multi-target synapses in insect brains based on a probabilistic graphical model. SyConn (Dorkenwald et al., 2017) contained a dedicated partner detection step subsequent to the voxel-based synapse detection, which classified contacts between segments from a local segmentation around neuron skeletons using the overlap with the voxel-based synapses and geometrical features. Huang et al. (2016) also included a dedicated partner detection step that consisted of classifying interfaces between identified presynaptic segments containing a T-bar and neighboring segments. The interfaces were defined as the overlap of two segments dilated by varying amounts and were thus neither restricted to the actual processes nor did they consider the pre- and postsynaptic process separately. More recently, additional methods for partner detection have been developed. Buhmann et al. (2018) proposed a method for partner detection directly from raw EM data using a 3D encoder-decoder type CNN to predict pairs of voxels in the pre- and postsynaptic processes of synapses. Parag et al. (2018) used a dedicated partner detection step to prune synapse candidates inferred from the voxelwise prediction of signed proximities that were overlapped with a volume segmentation. For candidate pruning, the raw EM data, the output of the voxelwise prediction of signed proximities and binary segmentation masks of two neighboring processes from a volume segmentation were used as input to a 3D CNN.

3.3. SynEM: Synapse Detection by Interface Classification

3.3.1. Interface Definition and Feature Representation

In SynEM, the task of synapse detection is formulated as a binary classification of interfaces between neuronal processes (see also Mishchenko et al., 2010; Kreshuk et al., 2014; Huang and Plaza, 2014; Dorkenwald et al., 2017). Neuronal processes are represented by a volume segmentation $S : \mathbb{Z}^3 \to \mathcal{L}$ of the image data, i.e. a partitioning into regions that represent processes from different cells which are separated by a one-voxel thick boundary (see also subsection 2.2.3). Boundary voxels are assigned the distinguished id 0 while segments have strictly positive ids. The border surfaces between two segments are given by the connected components of all boundary voxels that contain both segments in their 26 neighborhood N_{26} (see also Equation 2.4), i.e. for two segments with ids i and j the border surfaces $B(i, j)$ are given by

$$B(i,j) = CC_{26}(\{x \in S^{-1}(0) \mid i \in S(N_{26}(x)) \; and \; j \in S(N_{26}(x))\}), \tag{3.1}$$

where CC_{26} refers to the set of connected components of a set of voxels using the 26 neighborhood. The single connected components of a border are denoted by $b(i,j)^k \subset \mathbb{Z}^3,\ k = 1,\ldots,|B(i,j)|$ or simply $b(i,j)$ if the specific border surface does not matter. Thus, each border is associated with exactly two segments. Segments that have a non-empty border are called adjacent. The region adjacency graph, abbreviated with supervoxel graph, consists of all segments of the volume segmentation with edges between adjacent segments.

Borders between segments in a volume segmentation encode all available information about possible locations and orientations of synaptic junctions assuming that there are no merger errors at synapse locations. Furthermore, classifying borders as synaptic or non-synaptic includes both synapse detection as well as partner identification. However, the main features of synapses are not only located at borders, that is at the membranes between processes, but extend into the pre- and postsynaptic process: a vesicle cloud in the presynaptic terminal extending at least $100\,\mathrm{nm}$–$200\,\mathrm{nm}$ away from the membrane and a postsynaptic PSD with a width of about $20\,\mathrm{nm}$–$30\,\mathrm{nm}$. The regions in segments close to borders are included by defining subvolumes that extend up to a maximal distance from a border. For a border $b(i,j)$ and a maximal distance d, the subvolume for supervoxel i is defined by

$$sub_d(i) = \{x \in S^{-1}(i) \mid dist_{nm}(x, b(i,j)) \leq d\} \subset \mathbb{Z}^3, \tag{3.2}$$

where the distance is measured in nanometers to account for the data anisotropy. The interface $I(i,j)$ of a border $b(i,j)$ is defined as an ordered tuple of interface volumes $v_n \subset \mathbb{Z}^3$, i.e. sets of voxels, consisting of the border itself and $2k \in \mathbb{N}$ subvolumes with distances $d_1,\ldots,d_k$

$$I(i,j) = (b(i,j), sub_{d_1}(i), sub_{d_1}(j), \ldots, sub_{d_k}(i), sub_{d_k}(j)). \tag{3.3}$$

The ordering of subsegments of an interface is called the interface direction. For each segment of a border, three subvolumes with distances of $40\,\mathrm{nm}$, $80\,\mathrm{nm}$ and $160\,\mathrm{nm}$ were used to account for synaptic features at various distances from the membrane (Figure 3.3). Presynaptically, all three subvolumes are expected to be filled with vesicles, in particular also the $40\,\mathrm{nm}$ subvolume where vesicles fuse to the active zone, which is not the case for incidental non-synaptic touches, i.e. for locations where a bouton touches another process but without forming a synapse. Postsynaptically, the $40\,\mathrm{nm}$ subvolume contains the PSD, while larger subvolumes can help to discern the process identity, for example spine heads are typically mostly void of any ultrastructure making a synapse very likely while the presence of vesicles in the postsynaptic process indicates an axon and thus eliminates the possibility of a synapse. Interface classification thus contains border classification as a special case when no subvolumes are considered but allows one to take additional context around borders into account. Note that while the definition of a border is symmetric, i.e. $b(i,j) = b(j,i)$ this is not true for interfaces anymore due to the ordering of the subsegments in the interface definition. However, since synapses naturally have a direction from the presynaptic to the postsynaptic process, two versions of binary interface classification were considered which were called directed and undirected. For undirected interface classification, binary labels (synaptic or non-synaptic) were assigned to interfaces irrespective of the order of segments i and j in the

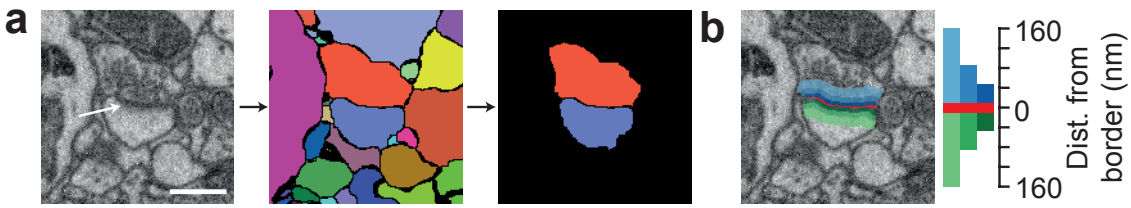

Figure 3.3: Interface definition. (a) The raw data (left, here from the dataset 2012-09-28_ex145_07x2; Boergens and Helmstaedter, 2012b) is volume segmented (middle, here using SegEM; Berning et al., 2015) and neighboring segments are identified (right). (b) For each pair of adjacent segments, the border (red) and the subvolumes with maximal distances of 40 nm, 80 nm and 160 nm (green and blue) are calculated. Scale bar: 500 nm (a-b). (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 2 `https://doi.org/10.7554/eLife.26414.005` licensed under CC BY 4.0)

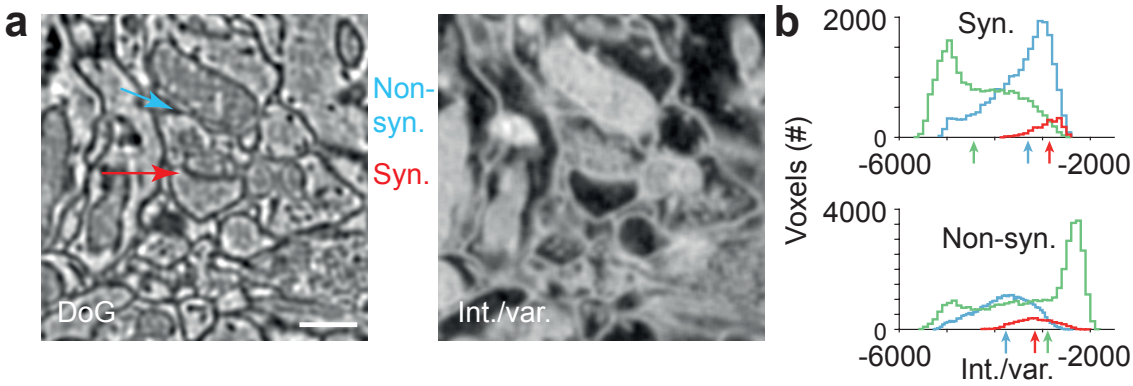

Figure 3.4: Texture feature calculation. (a) Filter responses of the Difference of Gaussians (DoG) and intensity/variance (Int./Var) filter calculated on the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b). Arrows point to examples of a synaptic (red) and a non-synaptic (blue) interface. (b) Distribution of the filter responses for the interface border (red) and largest subvolumes (green and blue; see also Figure 3.3) separately for the synaptic and non-synaptic interface highlighted in (a). Median responses for the subvolumes are indicated on the x-axis (arrows). Scale bar: 500 nm (a). (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 2 `https://doi.org/10.7554/eLife.26414.005` licensed under CC BY 4.0)

interface definition. This also offers the obvious data augmentation strategy of considering both directions of an interface with the same label during training. For directed interface classification, the synapse label is assigned to an interface $I(i, j)$ only if i corresponds to the presynaptic process and j to the postsynaptic process and a non-synaptic label otherwise.

For classification, a feature representation of interfaces was developed based on texture features as well as shape features. Texture features were calculated by applying 3D image filters to the raw data and pooling the filter response over the interface volumes using summary statistics. A set of 9 summary statistics was used consisting of quantiles (0, 0.25, 0.5, 0.75 and 1 quantile) and (standardized) moments (mean, variance, skewness and kurtosis) to describe the shape of the filter response distribution over the subvolumes (for an illustration see Figure 3.4). The texture features are summarized in subsection 3.3.2 and Table 3.1. In brief, eleven image filters were used that capture different textural properties of the raw data like edge detectors and band-pass filters: The identity filter (raw data), eigenvalues of the Structure Tensor and the Hessian matrix, Gaussian smoothing filter, Difference of Gaussians (DoG), Laplacian of Gaussian (LoG), Gauss Gradient magnitude, local standard deviation, intensity/variance feature, local entropy and a sphere average (see also Kreshuk et al., 2011; Becker et al., 2013). Shape features were calculated for the point

clouds of voxel locations in the interface subvolumes. To account for data anisotropy, the voxel locations were scaled by the voxel size of the data. Five shape features were calculated based on the interface volume, the principal axes and the convex hull of the border surface and the largest subvolumes (subsection 3.3.2 and Table 3.2). The feature vectors from the single texture and shape features were concatenated to give the overall interface feature vector of length 3224. A more detailed definition of the texture and shape features is given in subsection 3.3.2.

3.3.2. Detailed Feature Definition

The proposed set of interface texture features consisted of seven image filters based on Kreshuk et al. (2011) and Becker et al. (2013) and four additional filters. Each filter F_θ was applied to an input image stack I and outputs a response for each voxel $F_\theta I : \mathbb{Z}^3 \to \mathbb{R}^c$, where c specifies the number of filter channels and θ denotes the filter parameters. For each filter channel $F_\theta^i I : \mathbb{Z}^3 \to \mathbb{R},\ i = 1, \ldots, c$ the filter response was evaluated for all voxels in each subvolume v of an interface I and pooled using 9 summary statistics

$$\text{sum\_stats}\{F_\theta^i I(x) \mid x \in v\}. \tag{3.4}$$

A set of 5 quantiles (0, 0.25, 0.5, 0.75, 1) and 4 moments (mean, variance skewness, kurtosis) was used as summary statistics resulting in a feature vector of length $7 \cdot 9 = 63$ (number of subvolumes times number of summary statistics) for each filter channel. Different sets of parameters for a filter were treated as independent channels. The total number of texture features for a single interface was 3213 (see also Table 3.1).

Filters were calculated on patches of image data around an interface large enough such that no boundary effects, for example due to convolutions, were present at any interface voxel. The definition of the filters below is based on Staffler et al. (2017a). In the following $I * w$ denotes the discrete convolution (see Equation 2.6) of a 3D image stack I with a 3D convolution kernel w. The Gaussian convolutional kernel g_σ with standard deviation $\sigma = (\sigma_x, \sigma_y, \sigma_z) \in \mathbb{R}^3$ and a filter size $f = (f_x, f_y, f_z) \in \mathbb{N}_{>0}^3$ was defined by evaluating the unnormalized 3D Gaussian density function

$$\hat{g}_\sigma(x, y, z) = \exp\left(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2} - \frac{z^2}{2\sigma_z^2}\right), \tag{3.5}$$

in a rectangular bounding box $(x, y, z) \in [-f_x, f_x] \times [-f_y, f_y] \times [-f_z, f_z] \cap \mathbb{Z}^3$ and normalizing the result by the sum over its elements

$$g_\sigma(x, y, z) = \frac{\hat{g}_\sigma(x, y, z)}{\sum_{(x', y', z') \in U} \hat{g}_\sigma(x', y', z')}. \tag{3.6}$$

Derivatives of Gaussian filters were defined as derivatives of the Gaussian density g_σ resulting in the convolutional kernels

$$\frac{\partial}{\partial x} g_\sigma(x, y, z) = g_\sigma(x, y, z) \frac{-x}{\sigma_x^2}, \tag{3.7}$$

$$\frac{\partial^2}{\partial x^2} g_\sigma(x, y, z) = g_\sigma(x, y, z) \left(\frac{x^2}{\sigma_x^2} - 1\right) \frac{1}{\sigma_x^2}, \tag{3.8}$$

Filter	**Parameters**	**Channels**	**N**
Identity (raw data)	n. a.	1 (1)	63
Eigenvalues of Structure Tensor	$(\sigma_w, \sigma_d) \in \{(s,s),(s,2s),$ $(2s,s),(2s,2s),(3s,3s)\}$	15 (3)	945
Eigenvalues of Hessian	$\sigma \in \{s, 2s, 3s, 4s\}$	12 (3)	756
Gaussian smoothing	$\sigma \in \{s, 2s, 3s\}$	3 (1)	189
Difference of Gaussians (DoG)	$(\sigma, k) \in \{(s,1.5),(s,2),$ $(2s,1.5),(2s,2),(3s,1.5)\}$	5 (1)	315
Laplacian of Gaussian (LoG)	$\sigma \in \{s, 2s, 3s, 4s\}$	4 (1)	252
Gauss gradient magnitude	$\sigma \in \{s, 2s, 3s, 4s, 5s\}$	5 (1)	315
Local standard deviation	$U = 1_{5\times5\times5}$	1 (1)	63
Intensity/variance	$U = \{1_{3\times3\times3}, 1_{5\times5\times5}\}$	2 (1)	126
Local entropy	$U = 1_{5\times5\times5}$	1 (1)	63
Sphere average	$r \in \{3, 6\}$	2 (1)	126
Total			3213

Table 3.1: **SynEM texture features overview.** Filters used for texture feature calculation as described in subsection 3.3.1 were calculated for the specified parameter sets. For filters involving Gaussian kernels, the standard deviation in voxel space is specified in terms of $s \in \mathbb{R}^3$ with $s_i = \frac{12}{\text{voxelSize}_i}$, $i = 1, 2, 3$, and the corresponding filter size was set to $\frac{\sigma}{s}\text{ceil}(2s)$. Neighborhoods or structuring elements are denoted by $1_{x\times y\times z}$ referring to a matrix filled with ones of size $x \times y \times z$ in the respective dimension. Channels: Total number of filter channels used for summary statistic calculation including the number of different parameters (number of channels for each parameter is specified in parenthesis). N: All channels of a feature are separately pooled over the 7 interface volumes and summarized by 9 summary statistics resulting in 63 features per channel as described in subsection 3.3.1 summing up to a total number of N features for the corresponding filter.

and

$$\frac{\partial}{\partial x}\frac{\partial}{\partial y}g_\sigma(x,y,z) = g_\sigma(x,y,z)\frac{xy}{\sigma_x^2\sigma_y^2}, \tag{3.9}$$

and analogously for the other partial derivatives. Gaussian derivatives of images were defined as the convolution with the corresponding derivative of the Gaussian kernel and are denoted by

$$I_x^\sigma(x,y,z) = I * \frac{\partial}{\partial x}g_\sigma(x,y,z), \tag{3.10}$$

and

$$I_{xy}^\sigma(x,y,z) = I * \frac{\partial^2}{\partial x \partial y}g_\sigma(x,y,z), \tag{3.11}$$

and analogously for the other partial derivatives. The partial Gauss derivatives can be summarized as a vector-valued filter called the Gauss Gradient given by

$$\nabla_\sigma I = (I_x^\sigma, I_y^\sigma, I_z^\sigma) \tag{3.12}$$

which outputs a 3D vector for each voxel.

In the following let I denote a 3D raw data volume and $\sigma \in \mathbb{R}^3_{>0}$ the standard deviation for a Gaussian kernel. The Gaussian smoothing filter was defined as

$$F_\sigma^{\text{Gaussian}} I = I * g_\sigma \tag{3.13}$$

corresponding to a low-pass filter resulting in a denoised or blurred version of the original image.

The DoG filter was defined as

$$F_{\sigma,k}^{\text{DoG}} I = I * g_\sigma - I * g_{k\sigma}, \tag{3.14}$$

where the standard deviation of the second Gaussian filter results from the pointwise multiplication of σ with a positive scalar $k > 0$.

The Gauss gradient magnitude filter was defined as

$$F_\sigma^{\text{GaussGradMag}} I = \|\nabla_\sigma I\|_2 = \sqrt{(I_x^\sigma)^2 + (I_y^\sigma)^2 + (I_z^\sigma)^2}, \tag{3.15}$$

where the sum over images is defined pointwise.

The Structure Tensor $S = (S_{ij})_{i,j=1}^3$ was defined as a matrix valued filter with output components

$$S_{ij}(I) = (I_i^{\sigma_d} I_j^{\sigma_d}) * g_{\sigma_w}, \tag{3.16}$$

where $I_i^{\sigma_d}$ denotes the Gaussian derivative image of image I in the i-th dimension, σ_d is the standard deviation of the Gaussian derivative and σ_w is the standard deviation of a Gaussian

density filter called the window function. The multiplication of images is defined pointwise. Using the Gauss gradient, the structure tensor can be written in the compact form

$$S(I) = \nabla_\sigma I (\nabla_\sigma I)^T. \tag{3.17}$$

This shows that the Structure Tensor is a positive symmetric matrix which thus has three real valued and positive eigenvalues which allow the definition of the eigenvalues of the Structure Tensor filter given by

$$F^{\text{EVsStructTensor}}_{\sigma_d,\sigma_w} I = eig(S_{\sigma_d,\sigma_w}(I)), \tag{3.18}$$

where eig denotes the eigenvalues of S sorted in increasing order of their absolute value. To ensure exact symmetry only the lower triangular part of the Structure Tensor was calculated.

The LoG filter was defined as

$$F^{\text{LoG}}_\sigma I = I^\sigma_{xx} + I^\sigma_{yy} + I^\sigma_{zz}. \tag{3.19}$$

The Hessian matrix $H = (H_{ij})^3_{i,j=1}$ was defined as the second Gauss derivatives of an image

$$H_{ij}(I) = I^\sigma_{ij}, \tag{3.20}$$

where $I^{\sigma_d}_{ij}$ denotes the second Gaussian derivative image of image I in the i-th and j-th dimension. The Hessian is symmetric if the partial derivatives commute and thus has real eigenvalues. To enforce that H is symmetric only the lower triangular entries of H were calculated and the upper triangular entries were set to the corresponding entries from the lower triangular part. The eigenvalues of the Hessian matrix filter were defined as

$$F^{\text{EVsHessian}}_\sigma I = eig(H_\sigma(I)), \tag{3.21}$$

where eig denotes the eigenvalues of S sorted in increasing order of their absolute value as above.

The local entropy filter was defined as the entropy of voxel intensities $L \in \{0, \dots, 255\}$ in a neighborhood $U(p) \subset \mathbb{Z}^3$ around a voxel $p \in \mathbb{Z}^3$, i.e.

$$F^{\text{LocEnt}}_U I(p) = -\sum_{L \in \{0,\dots,255\}} p_{U(p)}(L) \log_2 p_{U(p)}(L), \tag{3.22}$$

where $p_U(L)$ is the relative frequency of voxel intensity $L = L(I)$ of image I in the neighborhood U.

The local standard deviation filter was defined as the standard deviation for all voxels in a neigh-

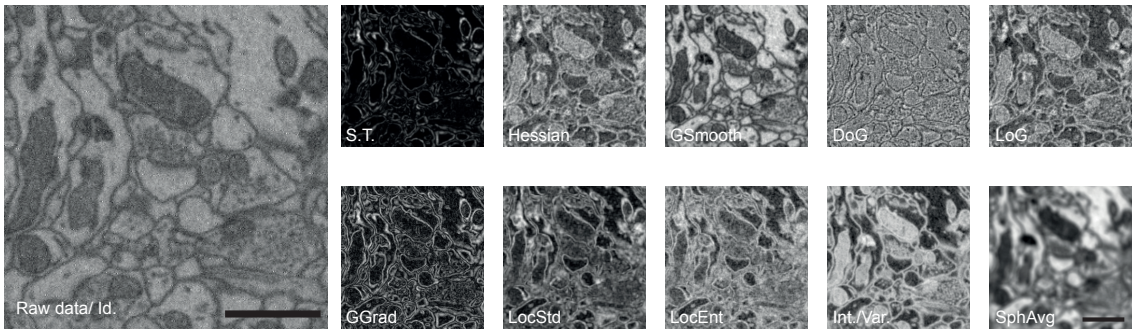

Figure 3.5: Texture filter examples used for interface feature calculation. All examples were calculated from the dataset dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b). Left: Identity filter corresponding to the raw data. Top row from left to right: Largest Eigenvalue of Structure Tensor (S.T.), Largest Eigenvalue of Hessian (Hessian), Gaussian smoothing filter (GSmooth), Difference of Gaussians filter (DoG), Laplace of Gaussian filter (LoG). Bottom row from left to right: Gauss gradient magnitude filter (GGrad), local standard deviation (LocStd), local entropy (LocEnt), intensity/variance filter (Int./Var.), sphere average (Sph.avg). Scale bars: 1 µm.

borhood $U(p) \subset \mathbb{Z}^3$ around the voxel of interest $p \in \mathbb{Z}^3$

$$F_U^{\text{LocStd}} I(p) = \sqrt{\frac{1}{|U(p)|-1} \sum_{p' \in U(p)} I(p') - \frac{1}{|U(p)|(|U(p)|-1)} \left(\sum_{p' \in U(x)} I(p) \right)^2}, \tag{3.23}$$

where $|U|$ denotes the number of voxels in U.

The intensity/variance filter was defined similar to the local standard deviation but without the normalization factors

$$F_U^{\text{Int/var}} I(p) = \sum_{p' \in U(p)} I(p') - \left(\sum_{p' \in U(p)} I(p') \right)^2. \tag{3.24}$$

The sphere average filter was defined as the mean of all voxels in a spherical neighborhood U_r around the voxel (x, y, z) of interest given by

$$U_r = \left\{ (x, y, z) \in \mathbb{Z}^3 \mid x^2 + y^2 + (2z)^2 \leq r^2 \right\}, \tag{3.25}$$

and hence

$$F_r^{\text{SphereAvg}} I = \frac{1}{|U_r|} \sum_{(x',y',z') \in U_r} I(x', y', z'). \tag{3.26}$$

An exemplary output for each texture filter is displayed in Figure 3.5.

Five shape features were calculated for the points clouds given by the interface volumes $v \subset \mathbb{Z}^3$ (Table 3.2). For anisotropic data, the voxel grid coordinates were converted to physical units by multiplying each voxel location by the voxel size $s \in \mathbb{R}^3$ resulting in the scaled point clouds $v^s = \{x \cdot s \mid x \in v\}$, where v is an interface volume and $\cdot$ denotes the componentwise multiplications of vectors. Due to the redundancy in the shape information for different subvolumes, only the

Feature	Subsegments	N
Volume	Bo., 160	3
Diameter	Bo.	1
Principal axis length	Bo.	1
Principal axes product	160	1
Convex hull	Bo., 160	3
All		11

Table 3.2: **SynEM shape features overview.** Feature: Name of the shape feature. Subsegments: Shape features for the interface feature representation were calculated for the interface border (Bo) and the two largest subvolumes (160). N: Length of the feature vector for the corresponding shape features.

interface border and the largest subvolumes were used. The volume feature was defined as the number of voxels in a subvolume $F^{Volume}(v) = |v|$ and was calculated for the border and largest subvolumes. The diameter feature was defined as the radius of a sphere with the same volume as an interface volume v, i.e.

$$F^{Diameter}(v) = \sqrt[3]{6\pi|v|\text{prod}(s)}, \tag{3.27}$$

where $\text{prod}(s)$ denotes the volume of a voxel given by the product of the components of the voxel size s, and was calculated for the border volume. The principal axes length feature was defined as the eigenvalues of the covariance matrix for the points in v^s and calculated for the border volume. The principal axis product feature was defined as the scalar product of the first principal components for the two largest subvolumes of an interface corrected for anisotropy. The convex hull feature was defined as the volume of the convex hull without the anisotropy correction and was calculated for the border and largest subvolumes. The resulting feature vector for shape features had a dimensionality of eleven.

The calculation of texture and shape features was implemented in Matlab. All filters involving Gaussians kernels or derivatives of Gaussian kernels were implemented as separable filters, i.e. instead of convolving the image data with a 3D convolutional kernel it is convolved consecutively by 3 1D kernels along different axes of the data (see also subsection 2.2.2). The eigenvalue calculation was done using an explicit formula for real symmetric 3×3 matrices implemented as a Matlab mex-function based on an implementation from the C++ matrix library Eigen. The local standard deviation filter and the local entropy filter were calculated using Matlab functions (`entropyfilt` and `stdfilt`) and the local standard deviation was later replaced by a custom separable implementation. The convex hull feature was calculated using the Matlab `convhull` function.

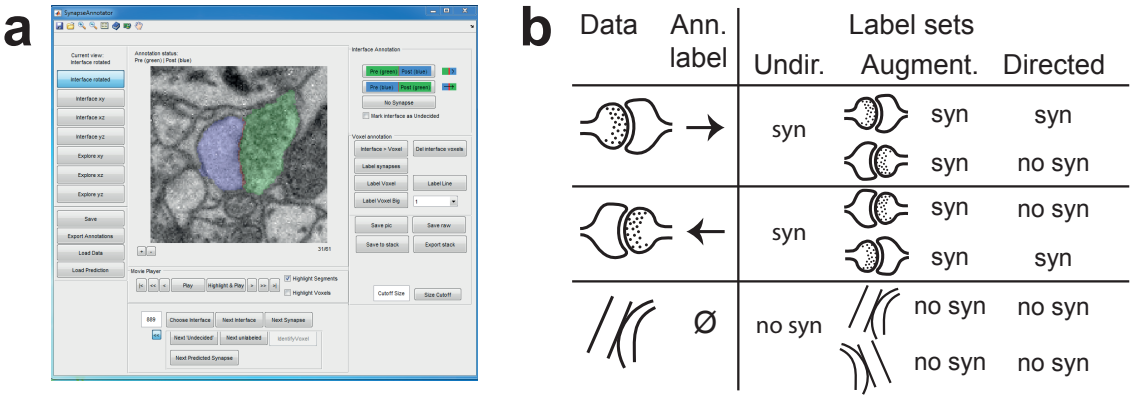

Figure 3.6: Training data annotation. (a) Matlab graphical user interface for neuronal interface annotation used for training and validation data generation (see subsection 3.3.3 for a detailed description). (b) Interface labeling strategies based on the subsegment ordering of interfaces exemplified for three different interfaces (rows). Human interface annotations consist of the identity of interfaces (synaptic, non-synaptic) as well as the synapse direction in terms of the subsegment ordering (Ann. label). The undirected label set (Undir.) uses one arbitrary interface orientation and assigns it with the interface identity (synaptic, non-synaptic). The augmented label set (Augment.) considers both orientations of an interface each with the same label as the undirected label set. The directed label set (Directed) also considers both orientations but only labels the direction pre- to postsynaptic subsegment of synapses as synaptic and the inverse direction as non-synaptic. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 3 `https://doi.org/10.7554/eLife.26414.008`, Figure 3 - Figure supplement 1 `https://doi.org/10.7554/eLife.26414.010` licensed under CC BY 4.0)

3.3.3. Classifier Training

The interface feature representation consisting of texture and shape features developed in the previous section was used to classify interfaces as synaptic or non-synaptic. For classification, an ensemble of boosted decision trees was trained using AdaBoostM1 (Freund and Schapire, 1997) or LogitBoost (Friedman et al., 2000) on a training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ of annotated interfaces. Decision trees were restricted to a depth of one (decision stumps), i.e. each weak learner consists of a threshold for a single dimension of x. For classifier training, the implementations of AdaBoostM1 and LogitBoost from the Matlab Statistical Toolbox (`fitensemble`) were used with a learning rate of 0.1, a total number of 1500 weak learners and a misclassification cost of 100 for the synaptic class.

For training data annotation, a custom Matlab graphical user interface was implemented (Figure 3.6a) that allows for direct one-click annotation of interfaces sequentially for all interfaces in a data volume. The user is presented an interface as a video (image stack) with the option to highlight the border and both subsegments in different colors (red, green, blue) centered on the center of mass of the interface border. For a consistent display of interface orientation, the raw data was rotated such that the second and third principal components of the interface border in a small window of $15 \times 15 \times 7$ voxels around its center of mass define the axes of the displayed image. Each interface is annotated with its identity (synaptic, non-synaptic) and in case of a synaptic interface also its direction by specifying which subsegment in the current orientation corresponds to the presynaptic process, for example green to blue. The human annotation labels (see Ann. label in Figure 3.6b) were used to define different label sets for interfaces as described above (subsection 3.3.1). More precisely, consider an interface $I(i, j)$ and its inverse orientation $I(j, i)$. The undirected label set assigns the interface identity label to one of the two orientations and discards the inverse orientation, i.e. $I(i, j) \in \{0, 1\}$ but $I(j, i)$ is not part of the training

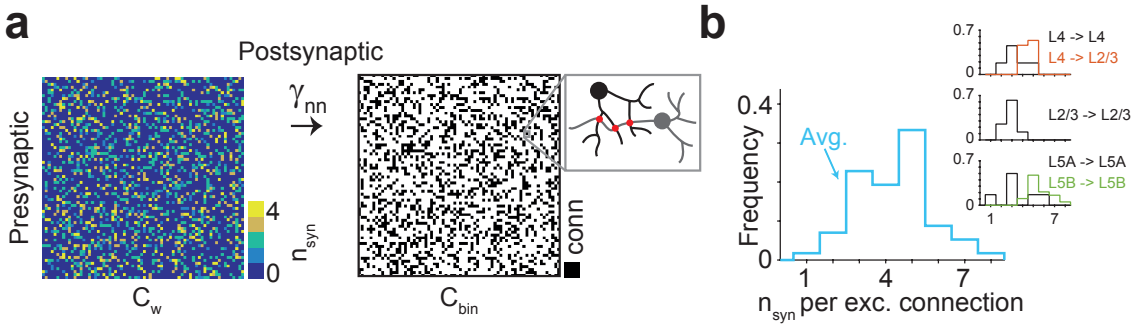

Figure 3.7: Cortical neuron-to-neuron connections. (a) A weighted connectome that reports the number of synapses per neuron-to-neuron connection (left) is transformed into a binary connectome (right) by considering only neuron pairs with at least γ_{nn} synapses as connected. (b) Distribution of the number of synapses between connected excitatory neuron pairs obtained from paired recordings in rodent cerebral cortex (Feldmeyer et al., 1999, 2002, 2006; Frick et al., 2008; Markram et al., 1997). (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 5 `https://doi.org/10.7554/eLife.26414.019` licensed under CC BY 4.0)

set. The augmented label set uses the interface identity to label both directions of an interface $I(i,j) = I(j,i) \in \{0,1\}$. The directed label set incorporates the synapse direction for synaptic interfaces by assigning interfaces with a synaptic label $I(i,j) = 1$ only if the interface is synaptic and i is the presynaptic segment and as non-synaptic otherwise. The different label sets are illustrated in (Figure 3.6b).

3.3.4. Connectome Error Estimation

For the connectomic reconstruction of a dataset, the synapses and corresponding pre- and postsynaptic processes that are detected by SynEM are combined with a neurite reconstruction to yield the weighted connectome of a circuit containing the number of synapses between pairs of neurons. For some applications such as network analysis, a binary connectome containing only the information whether two neurons are connected or not is already sufficient. Binary connectomes can easily be obtained from weighted connectomes by considering neurons with at least $\gamma_{nn} \in \mathbb{N}$ synapses as connected (Figure 3.7a). Neurons in mammalian cortex are known to establish synaptic connections via multiple synapses (Feldmeyer et al., 1999, 2002, 2006; Frick et al., 2008; Markram et al., 1997, 2004; Gupta et al., 2000; Hoffmann et al., 2015; Koelbl et al., 2015). Thus, if only the binary connectivity information is of interest, it is enough to detect only a subset of all synapses connecting two neurons in order to recover the neuron-to-neuron connection. This was formalized by estimating the neuron-to-neuron connectivity precision P_{nn} and recall R_{nn} based on the single synapse detection precision P_s and recall R_s measured on the test set.

For the estimate of neuron-to-neuron connectivity recall R_{nn}, the probability of detecting at least γ_{nn} synapses between a pair of neurons was estimated using a binomial model with a success probability equal to the single synapse recall R_s and a distribution $p(n)$ over the number of synapses between connected neurons

$$R_{nn} = P(k \geq \gamma_{nn} \mid R_s) = \sum_{n>0} Bin(k \geq \gamma_{nn} \mid n, R_s) p(n). \tag{3.28}$$

The distribution $p(n)$ for excitatory neurons was calculated by pooling the number of synapses

Publication	Source -> Target	No. cell pairs (No. syn.)
Feldmeyer et al. (1999)	L4 exc. -> L4 exc.	2(2), 5(3), 2(4), 2(5)
Feldmeyer et al. (2002)	L4 exc. -> L2/3 pyr.	6(4), 7(5)
Feldmeyer et al. (2006)	L2/3 exc. -> L2/3 pyr.	2(2), 5(3), 1(4)
Frick et al. (2008)	L5A pyr. -> L5A pyr.	1(1), 3(3), 1(5), 1(6)
Markram et al. (1997)	L5B pyr. -> L5B pyr.	2(4), 9(5), 4(6), 3(7), 1(8)
-	combined	1(1), 4(2), 13(3), 11(4), 19(5), 5(6), 3(7), 1(8)

Table 3.3: **Number of synapses between excitatory connected neurons.** Published studies of paired recordings about the number of synapses between connected neuron pairs. Middle column: Neuronal cell type and layer of the source and target neurons. Last column: Distribution over the number of synapses per connected neuron pair reported as the number of cell pairs (No. cell pairs) for the synapse count specified in parenthesis (No. syn.), i.e. 5(3) indicates that 5 pairs were found each connected by 3 synapses. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Supplementary File 2 `https://doi.org/10.7554/eLife.26414.032` licensed under CC BY 4.0)

between connected neuron pairs from published studies for excitatory neurons in rodent S1 (Feldmeyer et al., 1999, 2002, 2006; Frick et al., 2008; Markram et al., 1997) (Figure 3.7b and Table 3.3). For inhibitory neurons a fixed value of 6 was used based on Gupta et al. (2000); Hoffmann et al. (2015); Koelbl et al. (2015); Markram et al. (2004) (Table 3.4). Note that in this model false positive synapse detections do not contribute to the neuron-to-neuron recall.

To estimate the neuron-to-neuron connectivity precision P_{nn}, the false positive predictions made by a synapse classifier are distributed uniformly among all neuron-to-neuron connections. Consider a dataset with a total number of N neurons, N_{con} connected neurons pairs, N_{syn} synapses, an average number of synapses per connected neuron pair of $\overline{n_{syn}} = \frac{N_{syn}}{N_{con}}$ and a connectivity ratio of

$$c_r = \frac{N_{con}}{N^2}. \tag{3.29}$$

According to the definition of precision and recall, the number of false positive synapse detection can be estimated by

$$FP_s = \frac{1 - P_s}{P_s} R_s N_{syn}. \tag{3.30}$$

The false positive predictions are uniformly distributed among all N^2 possible neuron-to-neuron connections with a rate of

$$r_{FP} = \frac{FP_s}{N^2} = \frac{1 - P_s}{P_s} R_s \frac{N_{syn}}{N_{con}} \frac{N_{con}}{N^2} = \frac{1 - P_s}{P_s} R_s \overline{n_{syn}} c_r. \tag{3.31}$$

The number FP_{nn} of unconnected neuron-to-neuron pairs that get at least γ_{nn} false positive detections assigned was estimated by

$$FP_{nn} = N^2 (1 - c_r) Poi\left(x \geq \gamma_{nn} \mid r_{FP}\right), \tag{3.32}$$

Publication	Source -> Target	Average number of synaptic contacts
Koelbl et al. (2015)	L4 FS PV -> L4 exc. spiny	3.7 ± 1.3 (range $2 - 6$)
Hoffmann et al. (2015)	L2/3 inh. -> L2/3 pyr.	6.2 ± 2 (range $3 - 10$)
Gupta et al. (2000)	GABAergic F1	9.3 ± 3.1
	GABAergic F2	16 ± 5.5
	GABAergic F3	16.7 ± 11.9
Markram et al. (2004)	LBC -> pyr.	14.5 ± 1.7
	NBC -> pyr.	15.8 ± 4.1
	SBC -> pyr.	20.5 ± 10.5
	BTC -> pyr.	15.0 ± 7.1
	MC -> pyr.	11.2 ± 5.5

Table 3.4: **Number of synapses between inhibitory connected neurons.** Published studies of paired recordings about the number of synapses between connected neuron pairs. Middle column: Neuronal cell type of the source and target neuron populations. See corresponding publications for abbreviation of cell types. Last column: Average number of synapses per connection (mean $\pm$ std.) and range if available. (modified from Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Supplementary File 3 `https://doi.org/10.7554/eLife.26414.033` licensed under CC BY 4.0)

where $N^2(1 - c_r)$ is the number of unconnected neuron-to-neuron pairs and $Poi(x)$ refers to the Poisson distribution. Together with the number of true positive neuron-to-neuron connections

$$TP_{nn} = N^2 c_r R_{nn}, \tag{3.33}$$

this yields the neuron-to-neuron connection precision

$$P_{nn} = \frac{TP_{nn}}{TP_{nn} + FP_{nn}} = \frac{c_r R_{nn}}{c_r R_{nn} + (1 - c_r) Poi\left(x \geq \gamma_{nn} \mid r_{FP}\right)}. \tag{3.34}$$

Note that the neuron-to-neuron connectivity precision P_{nn} (Equation 3.34) does not depend on any absolute quantity (N, N_{con}, N_{syn}) anymore but only on the average number of synapses per connected neuron pair $\overline{n_{syn}}$ and the connectivity ratio c_r. For calculations in rodent S1 cortex the connectivity ratio was set to $c_r = 0.2$ for excitatory (Feldmeyer et al., 1999) and $c_r = 0.6$ for inhibitory connections (Gibson et al., 1999; Koelbl et al., 2015).

3.4. Experiments

The main SynEM evaluation was performed on a SBEM dataset from mouse S1 L4. The relevance of several design steps was quantified such as the interface subvolumes, the label sets and the feature representation. In addition, this dataset was used to compare the detection performance of

SynEM to previously published synapse detection methods. To examine the applicability of the SynEM approach to different volume EM data, SynEM was further evaluated on an ATUM-SEM dataset (Kasthuri et al., 2015) and compared to a state-of-the art approach for this dataset. The single synapse detection performance was used in the connectome error estimation framework developed in subsection 3.3.4 to estimate the neuron-to-neuron error, which is exemplified for a sparse connectome consisting of 104 axonal and 100 postsynaptic processes in subsection 3.4.5.

3.4.1. SBEM Dataset and Label Data Generation

The dataset used for the development and main evaluation of SynEM is a SBEM dataset from mouse S1 L4 of size $93 \times 60 \times 93\,\mu\mathrm{m}^3$ imaged at a voxel size of $11.24 \times 11.24 \times 28\,\mathrm{nm}^3$ (dataset 2012-09-28_ex145_07x2, Boergens and Helmstaedter, 2012b; see also Berning et al., 2015; Staffler et al., 2017a). In brief, the tissue was conventionally en-block stained (Briggman et al., 2011) without extra-cellular space preservation (see for example Pallotto et al., 2015). A centered volume of size $86 \times 52 \times 86\,\mu\mathrm{m}^3$ of the dataset was segmented using the SegEM segmentation algorithm (Berning et al., 2015). Membrane prediction was done using CNN $20130516T204040_{8,3}$. For training data generation, segmentation parameters optimized for whole cell segmentations were used ($r_{se} = 0$; $\theta_{ms} = 50$; $\theta_{hm} = 0.25$). For test data generation, the segmentation parameters $r_{se} = 0$; $\theta_{ms} = 50$; $\theta_{hm} = 0.39$ were used (see Berning et al., 2015, Table 2).

Training and validation labels for interface classification were generated for 40 non-overlapping rectangular volumes (39 of size $5.6 \times 5.6 \times 5.6\,\mu\mathrm{m}^3$ and one of size $9.6 \times 6.8 \times 8.3\,\mu\mathrm{m}^3$; Figure 3.8a). Interface volumes as defined in subsection 3.3.1 were extracted using subsegments with distances of $40\,\mathrm{nm}$, $80\,\mathrm{nm}$ and $160\,\mathrm{nm}$. Interfaces with a center of mass closer than $1.124\,\mu\mathrm{m}$ to the training cube border were discarded. Interface annotation was done using the custom Matlab graphical user interface described in subsection 3.3.3. The resulting label set consisted of $75,383$ interfaces out of which 1858 were synaptic. Eight label volumes comprising 391 synaptic and 11906 non-synaptic interfaces were used as validation set while the remaining 32 were used for training. If the directed label set was used for training then the predictions on the validation set for the different directions of an interface were combined using a logical OR operation. The feature calculation was done as described in subsection 3.3.2. For training of the voxel classification methods proposed by Kreshuk et al. (2011) and Becker et al. (2013), voxel labels for synaptic junctions and non-synaptic voxels were made using Ilastik (Sommer et al., 2011). Sparse voxel labels were made for five regions of size $3.4 \times 3.4 \times 3.4\,\mu\mathrm{m}^3$ in the center of five training volumes. Synaptic labels were made by annotating parts of the PSD of 115 synapses that were also annotated in the training cubes (note that Becker et al., 2013 only used 7-20 synapses during training). Non-synaptic labels were created iteratively by creating non-synaptic labels for two training cubes first and then adding non-synaptic labels for the remaining three training cubes based on misclassified locations using the interactive prediction mode in Ilastik. At the end, the non-synaptic labels in the first two training cubes were also extended based on misclassified locations resulting in roughly twice as many non-synaptic voxel labels than synaptic ones. For the object classification step in Kreshuk et al. (2014), the graph cut segmentation with label smoothing ($[1, 1, 0.5]$ voxel standard deviation), a voxel probability threshold of 0.5 and a graph cut

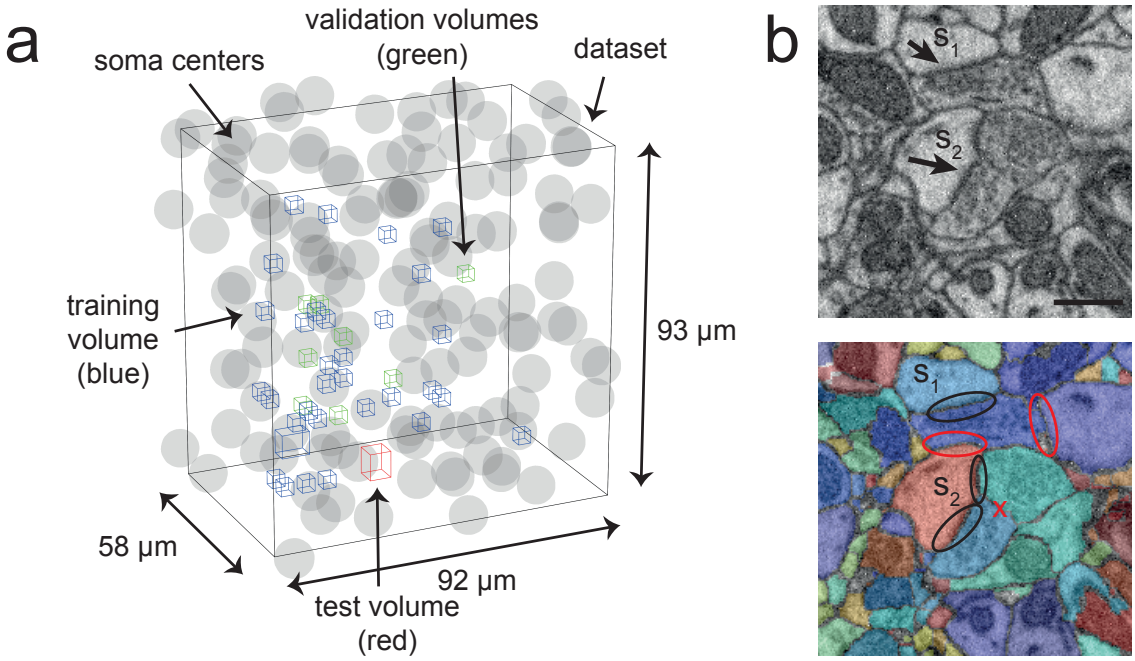

Figure 3.8: SynEM training and test data. (a) Spatial locations of training, validation and test volumes within the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) used for SynEM training and evaluation. (b) Top: Two synapses from the SBEM test set (s_1 and s_2 marked by arrows). Bottom: Overlay with the test set segmentation showing that synapse s_2 consists of two interfaces while s_1 consists of a single interface (black circles). Examples of non-synaptic interfaces are marked by red circles. Interfaces that overlap with a synapse PSD annotation but are not part of the synapse are not labeled as synaptic, for example the interface in the presynaptic axon of synapse s_1 marked by a red x. Scale bar: 500 nm (b). (a taken from: Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 3 - Figure Supplement 2 https://doi.org/10.7554/eLife.26414.011 licensed under CC BY 4.0)

constant of $\lambda = 0.5$ was used. Object detections were annotated in the center of five training cubes of size $3.4 \times 3.4 \times 3.4\,\mu m^3$ different form the cubes used for voxel prediction resulting in 299 labels (101 synaptic, 198 non-synaptic). The label cubes for training of the multi-class CNN approach SyConn (Dorkenwald et al., 2017) were made by annotating six out of the 32 training volumes densely with voxel labels for synaptic junctions, vesicle clouds and mitochondria resulting in a total labeled training volume of $225\,\mu m^3$. For vesicle cloud and mitochondria annotation, the predictions of a CNN trained on membranes, mitochondria and vesicle clouds were proofread by undergraduate students and twice by an expert annotator. An expert neuroscientist used the annotated synapses in the training cubes to add voxel labels for synaptic junctions.

For testing, a volume of size $5.75 \times 5.75 \times 7.17\,\mu m^3$ ($512 \times 512 \times 256$ voxels; bounding box $[3713, 2817, 129, 4224, 3328, 384]$) disjunct from all training and validation volumes was randomly selected such that it did not contain a soma or dominantly large dendrites. Synapse detection was done in a multi-step procedure involving three expert annotators. First, an expert neuroscientist did a volume search for synapses using webKnossos (Boergens et al., 2017). Subsequently, the same expert traced the axons for all detected synapses and inspected vesicle clouds along axons for further synapses. The previous step was repeated for any missed boutons that were found. In total, the first expert detected 335 potential synapse locations. Based on the annotations of the first expert, a second expert added additional 8 potential synapse locations resulting in a total number of 343 potential synapse locations. All potential synapse locations were again

independently annotated by both experts as synaptic and non-synaptic and the disagreeing locations were jointly resolved (282 were labeled as synaptic by each annotator out of which 261 were in agreement). After this procedure, a total of 278 synapses were identified with a precision and recall of the single expert annotators with respect to the consensus tracing of 93.6%, 94.6% (expert 1) and 97.9%, 98.9% (expert 2). Afterwards, expert 1 added voxel labels for all synaptic junctions (PSD). In addition, all shaft synapses were labeled by expert 1 and proofread by expert 2 resulting in 36 shaft synapses. All interfaces with a border that had any overlap with the voxel labels of one of the 278 synapses were extracted and labeled as synaptic or non-synaptic by expert 1. Interfaces closer than $160\,\mathrm{nm}$ to the test set boundary were discarded resulting in 235 remaining synapses out of which 31 were labeled as shaft synapses.

To determine inhibitory synapse detection performance for connectome error estimation, an additional test set with interfaces along inhibitory axons was created. Skeleton tracings of three inhibitory axons were used by the same two experts as above to detect all synapse locations along the axons. Disagreeing locations were jointly resolved by both annotators resulting in a test set of 171 synapses. All segments in a postsynaptic process close to a synapse location were annotated and all interfaces between the axon and the postsynaptic process were associated with the corresponding synapse.

After training, SynEM was applied to the whole segmented volume of the dataset. Interface calculation was done on non-overlapping cubes of size $512 \times 512 \times 256$ and the feature calculation on larger cubes of size $548 \times 548 \times 268$ voxels with an overlap of $72 \times 72 \times 24$ voxels in x, y and z-direction respectively to ensure that no feature boundary effects were present at any interface voxels. The synapse count for the whole dataset was obtained by clustering synaptic interfaces for the optimal single synapse threshold θ_s of the test set using hierarchical clustering with a distance cutoff of $320.12\,\mathrm{nm}$, which was optimized on the test set.

3.4.2. Synapse Detection Performance Evaluation

SynEM was evaluated for different label sets, feature representations and classifiers (Figure 3.9 and Table 3.5). The initial classifier used only the $160\,\mathrm{nm}$ interface subvolumes, the texture features described in Kreshuk et al. (2011), the volume shape feature, quantiles as summary statistics and was trained using AdaBoostM1 on the undirected label set resulting in an optimal F1 score of 0.734. Then, additional features (texture and shape), subvolumes ($40\,\mathrm{nm}$ and $80\,\mathrm{nm}$ subvolumes) and summary statistics (moments) were added increasing the optimal F1 score to 0.779. Substantial improvements were observed when either changing the classification algorithm from AdaBoostM1 to LogitBoost or using the directed label set resulting in F1 scores of 0.835 and 0.839 respectively. Training with both the directed label set and a LogitBoost classifier resulted in the best validation set performance with an F1 score of 0.865 (Figure 3.9a).

For the evaluation on the test set, a synapse was considered detected if at least one of the synaptic interfaces overlapping with the synapse was detected (TP). A synapse was considered missed if none of the overlapping interfaces were detected (FN). An interface prediction was considered a

Classifier	Subvols.	Features	Sum. stats.	Train. algo.	Label	F1
Init.	160	Identity, EVs of Structure Tensor and Hessian, Gauss. Smooth, LoG, Gauss Grad. Magn., Volume	quant.	AdaBoostM1	Undir.	0.734
Add feat.	160	all	quant.	AdaBoostM1	Undir.	0.753
Add subvol.	all	all	quant.	AdaBoostM1	Undir.	0.763
Add stats.	all	all	all	AdaBoostM1	Undir.	0.779
Direct	all	all	all	AdaBoostM1	Dir.	0.839
Logit	all	all	all	LogitBoost	Undir.	0.835
Augment & Logit	all	all	all	LogitBoost	Augm.	0.839
Direct & Logit	all	all	all	LogitBoost	Dir.	0.865

Table 3.5: **SynEM classifier development.** Hyperparameter choices for the classifiers compared on the validation set (see Figure 3.9). Subvols.: Subvolumes used in interface definition in nanometers. 'all' refers to $40\,\mathrm{nm}$, $80\,\mathrm{nm}$ and $160\,\mathrm{nm}$. Features: Features used in the interface feature representation (see Table 3.1 and Table 3.2). Sum. stats.: Summary statistics used for texture feature calculation. 'quant.' refers to 5 quantiles only while 'all' refers to all 9 summary statistics consisting of quantiles and moments. Train. algo.: Training algorithm used for classifier training. Label: Undirected (Undir.), augmented (Augm.) or directed (Dir.) label set used for training (see Figure 3.6). F1: Optimal F1 score on the validation set.

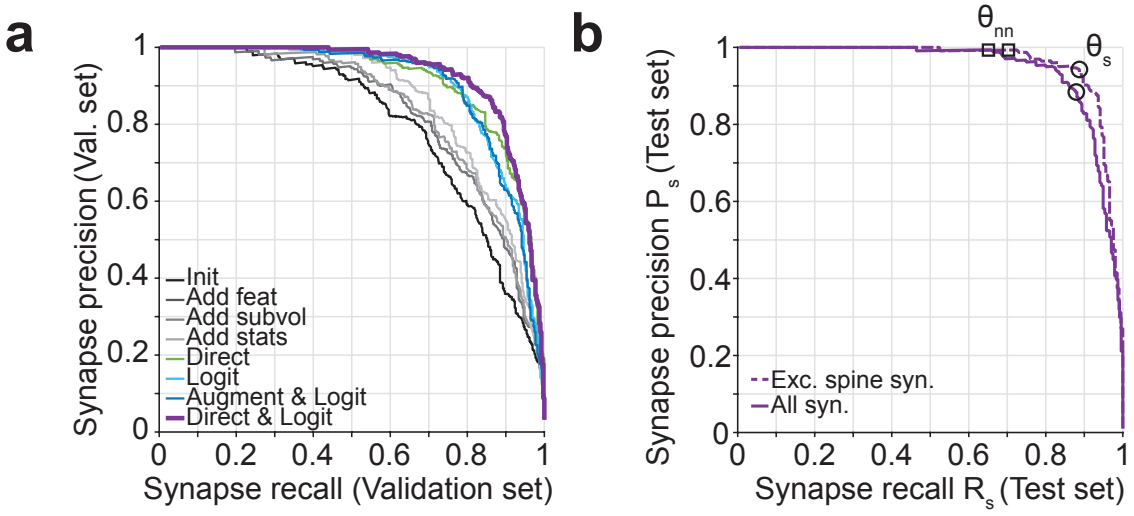

Figure 3.9: SynEM SBEM evaluation. (a) Validation set performance of SynEM on the 3D SBEM data for different features, aggregation statistics, classifier parameters (see Table 3.5) and label sets (see Figure 3.6). If not otherwise stated, the undirected label set and AdaBoostM1 was used. Classifier hyperparameters were set as described in subsection 3.3.3. Init: Initial classifier using only the 160 nm interface subvolumes, a subset of all features (Identity, Eigenvalues of Structure Tensor and Hessian, Gaussian smoothing, LoG, Gauss gradient magnitude, volume) and quantiles as summary statistics. The initial classifier was extended by using all features ('Add feat', see Table 3.1 and Table 3.2), 40 nm and 80 nm subvolumes ('Add subvol') and moments as additional summary statistics ('Add stats'). Direct: Classifier trained on the directed label set. Logit: Classifier trained using LogitBoost. Augment and Logit: LogitBoost classifier trained on the augmented label set. Direct and Logit: LogitBoost classifier trained on the directed label set. (b) Test set performance of SynEM on 3D SBEM data for the best classifier on the validation set (Direct & Logit) for all synapses (All syn., solid line) and for spine synapses only (exc. spine syn., dashed line). The SynEM interface thresholds for optimal single synapse detection performance (θ_s, black circle) and optimal connectome reconstruction performance (θ_{nn}, black square; see subsection 3.3.4) are listed in Table 3.7. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 3, legend modified, `https://doi.org/10.7554/eLife.26414.008` licensed under CC BY 4.0)

false positive detection if the interface did not overlap with any ground truth synapse (FP) (see also Figure 3.8b). Note that this evaluation strategy was different to the validation set, where each synaptic interface needed to be detected irrespective of whether another interface overlapping with the same synapse was detected as well. Thus, the test set evaluation strategy focused on the actual synapse detection performance and prevented FN detections at small interfaces overlapping with synapses, which was more frequent for the test set compared to the validation set due to the stronger oversegmentation.

The best classifier from the validation set ('Direct & Logit') achieved an optimal F1 score of 0.8839 and precision and recall rates of 88% on the test set. Restricted to spine synapses, the classifier performance was even better with an F1 score of 0.91 (94% precision and 89% recall). Classification examples at the threshold θ_s for optimal single synapse performance are shown in Figure 3.10a. Qualitatively, typical errors for false negative detections were small vesicle clouds, small PSDs or mitochondria in the pre- and/or postsynaptic process. False positive errors were due to vesicle clouds close membranes without an actual synaptic contact (Figure 3.10a FP) and mitochondria. Furthermore, segmentation errors were often found close to misclassified interface locations. To quantify the local segmentation quality, the FP and FN detection errors were inspected revealing that 26 out of 28 FNs and 22 out of 27 FPs were located close to segmentation errors. A local correction of these errors resulted in correct classification for 22 of the 48 cases (46%) indicating that improvements in volume segmentation quality could further improve

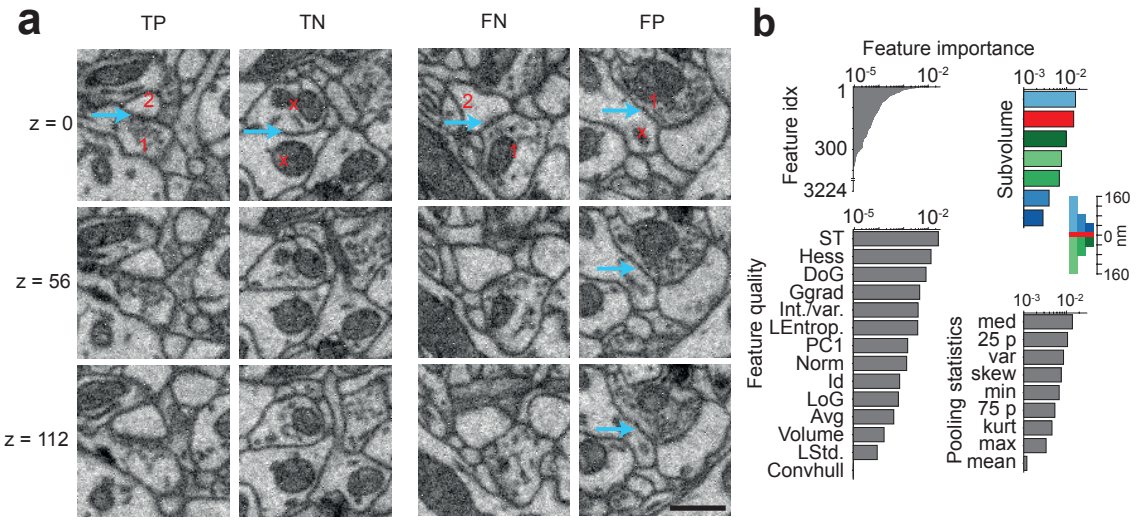

Figure 3.10: SynEM classification examples and feature importance. (a) Examples from the confusion matrix (TP, TN, FN, TP) for the SynEM classification from the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) at the optimal classification threshold θ_s (circle in Figure 3.9) shown in 3 image planes spaced by 56 nm, i.e. every second slice. Blue arrows indicate the classified interface. 1 and 2 mark the pre- and postsynaptic process respectively, while x marks processes that do not form a synapse at the displayed location. Note that the presynaptic process (1) in the FP example innervates a spine head but the FP detection was made onto another process (x) that is non-synaptic at this location. (b) Ranked importance of SynEM features determined from the tree-based classifier after training: All features (top left), relevance of image filters (bottom left), subvolumes used for pooling (top right) and summary/pooling statistics (bottom right). 378 out of 3224 features were relevant for the best performing classifier. Scale bar: 500 nm (a). (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 4, legend modified, `https://doi.org/10.7554/eLife.26414.016` licensed under CC BY 4.0)

SynEM performance.

The relevance of different features used for interface classification was assessed using that features in boosted decision trees can be ranked by their classification importance (Figure 3.10b). Out of all 3224 features, the classifier only used 378. Leaving out unused features does not impede classification accuracy and can thus be used to speed up prediction for large dataset. The 10 single most important features contained 2 image filters and a shape features that were not part of the initial classifier (Table 3.6). Furthermore, the corresponding features originated from all subvolumes using several different quantiles and moments. The total importance of image filters, subvolumes and summary statistics was calculated by aggregating the feature importance of all features for the corresponding category. This reveals that the most important subvolumes are the 160 nm presynaptic volume, the interface border and the 40 nm postsynaptic volume suggesting that the classifier was able to learn the asymmetry of synapses and utilize the different information content of the interface subvolumes.

Next, the performance of SynEM was compared to previously published synapse detection methods (Figure 3.11). Since each method is typically developed and evaluated on a different type of EM data, the published implementations of previously proposed synapse detection methods were used to train and evaluate classifiers on the SynEM test set (Figure 3.11). SynEM was compared to the approaches proposed by Kreshuk et al. (2011, 2014); Becker et al. (2013) and Dorkenwald et al. (2017) which were shown to be the best performing methods superior to Perez et al. (2014) and Neila et al. (2016). Methods developed for specific types of synapses (Jagadeesh et al., 2014;

Rank	Feature	Parameters	Subvol.	Sum. stat.
1	EVs of Struct. Tensor (largest)	$\sigma_w = 2s, \sigma_D = s$	160 nm, S1,	Median
2	EVs of Struct. Tensor (smallest)	$\sigma_w = 2s, \sigma_D = s$	160 nm, S1,	Median
3	Local entropy	$U = 1_{5\times5\times5}$	160 nm, S2	Variance
4	Difference of Gaussians	$\sigma = 3s, k = 1.5$	Border	25^{th} perc
5	Difference of Gaussians	$\sigma = 2s, k = 1.5$	Border	Median
6	EVs of Struct. Tensor (middle)	$\sigma_w = 2s, \sigma_D = s$	40 nm, S2	Min
7	Int./var.	$U = 1_{3\times3\times3}$	Border	75^{th} perc
8	EVs of Struct. Tensor (largest)	$\sigma_w = 2s, \sigma_D = s$	80 nm, S1	25^{th} perc
9	Gauss. gradient magnitude	$\sigma = s$	40 nm, S2	25^{th} perc
10	Principal axes length (2nd)	-	Border	-

Table 3.6: **SynEM feature importance.** Feature importance ranked by the boosted ensemble. For features see Table 3.1 and Table 3.2. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Table 2 https://doi.org/10.7554/eLife.26414.018 licensed under CC BY 4.0)

Plaza et al., 2014; Huang et al., 2016) were not considered. Note that due to the voxel-based nature of these approaches, the ground truth synaptic junction segmentation of the test set and voxel training data describe in subsection 3.4.1 were used instead of the SynEM interface annotations. The classifier training for Kreshuk et al. (2011, 2014); Becker et al. (2013) was done using the implementation provided by Ilastik (Sommer et al., 2011). For the voxel classification step in Kreshuk et al. (2011) and Kreshuk et al. (2014), all features up to a standard deviation of 5 voxels were used. For the object classification step in Kreshuk et al. (2014), all features provided by Ilastik were used. For the method by Becker et al. (2013), all features proposed by the authors and 200 weak learners were used. For classification, the test set was tiled into four cubes of size $256 \times 256 \times 256$ voxels ($2.9 \times 2.9 \times 7.2\,\mu\text{m}^3$) with a boundary of $280\,\text{nm}$ and the prediction was thresholded and morphologically closed using a cubic structuring element of 3 voxels side length. Only components with size larger than 50 voxels were kept. The recursive CNN training for SyConn (Dorkenwald et al., 2017) on synaptic junctions, mitochondria and vesicle clouds was done using same architectures and hyperparameters as described by the authors. Only the output channel for synaptic junctions was used for performance comparison and only connected components of synapse predictions with at least 250 voxels were kept. Synapse detection precision and recall rates for all methods were calculated based on the overlap of the predicted synapse segmentations with the ground truth synapse segmentation. Ground truth synapses that had any overlap with a predicted synapse were defined as TP detections and otherwise, if they did not overlap with any predicted synapse, as FN detections. To minimize boundary effects, predicted synapses at the center of the test set of size $484 \times 484 \times 246$ voxels that did not overlap with any ground truth synapse were defined as FP detections. The performance of SynEM in comparison

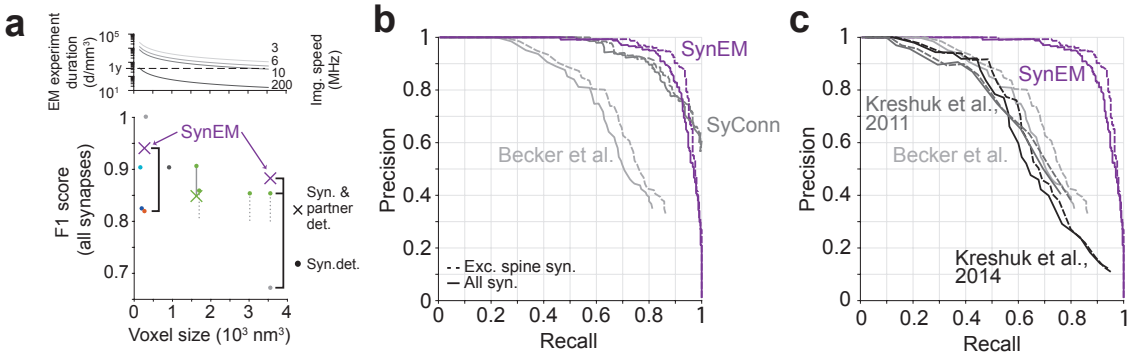

Figure 3.11: Synapse detection method comparison. (a) Top: Duration of a 3D EM experiment as a function of the imaging resolution and the imaging speed exemplified for a dataset of size $1\,\mathrm{mm}^3$. Note that for most current EM experimental setups with an imaging speed less than 10 MHz, experiments are only realistic for smaller voxel sizes. Bottom: Performance of published synapse detection methods reported as $F1$-score as a function of the voxel size: dark blue, (Mishchenko et al., 2010); cyan, (Kreshuk et al., 2011); light gray, (Becker et al., 2013); dark gray, (Kreshuk et al., 2014); red, (Roncal et al., 2015); green, (Dorkenwald et al., 2017); purple, SynEM; Direct performance comparison of SynEM with Roncal et al. (2015) on the ATUM-SEM dataset (Kasthuri et al., 2015) (see subsection 3.4.4) and SynEM with SyConn (Dorkenwald et al., 2017) on the SynEM dense test set. Note that the top and bottom plot share the same x-axis. (b) Precision-recall curves of SynEM, SyConn (Dorkenwald et al., 2017) and the method of Becker et al. (2013) on the SynEM test set for all synapses (solid lines) and excitatory spine synapses only (dashed line). Note that SyConn was shown to outperform all previously proposed synapse detection methods (Dorkenwald et al., 2017) while Becker et al. (2013) was shown to outperform Kreshuk et al. (2011) (see also panel c). (c) Precision-recall curves of SynEM, the methods of Kreshuk et al. (2011), Kreshuk et al. (2014) and Becker et al. (2013) on the SynEM test set all for synapses (solid lines) and for excitatory spine synapses only (dashed line). (a and b from Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 3, `https://doi.org/10.7554/eLife.26414.008` licensed under CC BY 4.0)

to the two best performing competing methods (Becker et al., 2013; Dorkenwald et al., 2017) is shown in Figure 3.11b. SynEM outperforms the other methods by a substantial margin except in a high recall regime where SyConn (Dorkenwald et al., 2017) is superior.

3.4.3. Biological Plausibility

To further validate the SynEM interface classification approach, the predicted synapse density of the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) was calculated. SynEM was applied to the whole dataset 2012-09-28_ex145_07x2 at the optimal single synapse threshold θ_s determined on the test set (see Figure 3.9 and Table 3.7). The runtime for the total volume of $384\,592\,\mu\mathrm{m}^3$ was 2.6 hours on a cluster with 480 CPU cores and 16 GB RAM per core implying a total runtime of 279.9 days for a dataset of size $1\,\mathrm{mm}^3$, which is orders of magnitude faster than human annotation (10^5 to 10^6 hours, see Figure 1.2) without any particular speed optimizations (plain Matlab code). The predicted synaptic interfaces were clustered to account for synapses that were split into multiple interfaces resulting in a total number of 195644 synapses for half of the dataset corresponding to a synapse density of 1.02 synapses per $\mu\mathrm{m}^3$, which is consistent with previous reports (Merchán-Pérez et al., 2014).

To account for systematic biases in synapse detection due to synapse size, the axon-spine interface area (ASI) of the ground truth spine synapses retrieved by SynEM on the test set was measured (Figure 3.12). For a spine synapse detected at the optimal single synapse threshold θ_s, the ASI was calculated as the sum of the area of all ground truth interfaces associated to the synapse resulting in an average ASI of $0.264 \pm 0.206\,\mu\mathrm{m}^2$ (mean $\pm$ s.d.; range $0.033\,\mu\mathrm{m}^2$–$1.189\,\mu\mathrm{m}^2$, $n = 181$). The spine synapse ASI was compared to the ASI distributions of de Vivo et al. (2017)

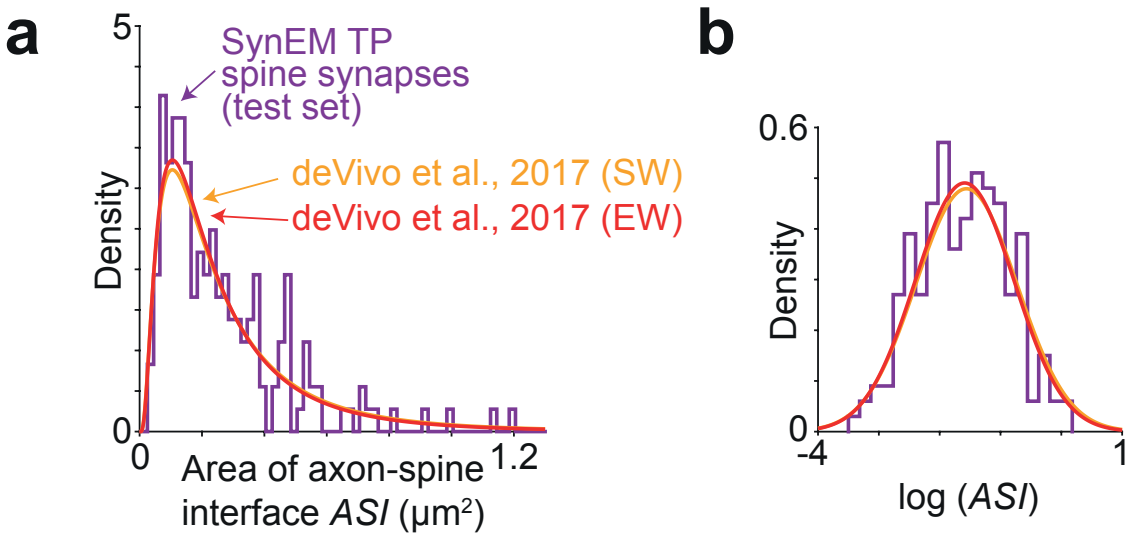

Figure 3.12: Synapse size comparison in SBEM data. (a) Histogram of the axon-spine interface area (ASI) of spine synapses in the SynEM test set from the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) detected by SynEM (purple) overlaid with the ASI distributions found in de Vivo et al. (2017) under the two wakeful conditions (SW: spontaneous wake, yellow; EW: enforced wake, red). Note that while both datasets are from mouse S1, the resolution of the SynEM test set was $11.24 \times 11.24 \times 28\,\mathrm{nm}^3$ while the dataset of de Vivo et al. (2017) had a resolution of $5.9\,\mathrm{nm}$ (xy plane) with a section thickness of $54.7 \pm 4.8\,\mathrm{nm}$ (SW) and $51.4 \pm 10.3\,\mathrm{nm}$ (EW). (b) The distributions of (a) shown on a natural logarithmic scale (log ASI SynEM: -1.60 ± 0.74, $n = 181$; log ASI SW: -1.56 ± 0.83, $n = 839$; log ASI EW: -1.59 ± 0.81, $n = 836$, mean $\pm$ s.d.). The distributions are indistinguishable ($p = 0.52$ for SynEM vs. SW and $p = 0.83$ for SynEM vs. EW; two-sample two-tailed t-test) indicating that SynEM does not have a bias depending on synapse size. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 7, legend modified, `https://doi.org/10.7554/eLife.26414.029` licensed under CC BY 4.0)

for the spontaneous wake (SW) and enforced wake (EW) conditions (de Vivo et al., 2017, Table S1) assuming that their ASI distributions are lognormal to convert the sample mean and standard deviation into the corresponding lognormal distribution parameters. Two-sample two-tailed t-tests were performed to compare the logarithmic values of the ASI of the SynEM test set detections with the lognormal distributions from de Vivo et al. (2017) ($\log$ ASI -1.56 ± 0.83, $n = 839$, SW; $-1.59 \pm 0.81, n = 836$, EW; mean $\pm$ s.d.) showing that the distributions are indistinguishable ($p = 0.5175$, SW; $p = 0.8258$, EW). This shows that synapses detected in lower resolution SBEM data (in-plane resolution of $11.24\,\mathrm{nm}$ with a section thickness of $26\,\mathrm{nm}$–$30\,\mathrm{nm}$ for 2012-09-28_ex145_07x2) yields a similar size distribution as higher resolution data (in-plane resolution of $5\,\mathrm{nm}$ with a section thickness of about $50\,\mathrm{nm}$ in de Vivo et al., 2017) indicating that reliable synapse detection in this data is possible and that SynEM has no obvious synapse detection bias depending on synapse size.

3.4.4. ATUM Dataset

To test the applicability of SynEM to volume EM data acquired using imaging techniques different from SBEM, a dataset from mouse S1 imaged using ATUM-SEM with a voxel size of $3 \times 3 \times 30\,\mathrm{nm}^3$ was used (Kasthuri et al., 2015). The raw data (kasthuri11cc), the volume segmentation (kat11segments) and synapse segmentation (kat11synapses) that are publicly available at openconnecto.me were downloaded at resolution 1 corresponding to $6 \times 6 \times 30\,\mathrm{nm}^3$ in the bounding box $[2432, 7552; 6656, 10112; 769, 1537]$ using the Matlab CAJAL API[1] . Note that the

[1] https://github.com/neurodata-arxiv/CAJAL

segmentation in this bounding box is not dense, i.e. not every single process in the bounding box is segmented. However, interface detection between neighboring segmented processes is still possible. Furthermore, the segmentation from Kasthuri et al. (2015) was created manually and does not have the one-voxel thick boundary between processes required in the SynEM definition of borders (cf. Equation 3.1). To post-hoc introduce a one-voxel boundary between processes, the segmentation was tiled into non-overlapping cubes of size $[1024, 1024, 512]$ and each segment was separately eroded by a cubic structuring element with 3 voxels side length. Then, a watershed segmentation was run on the distance transform of the eroded segmentation. Since the watershed produces a dense segmentation, unlabeled regions in the adapted segmentation were recovered by settings the ids of voxels with a segment id of 0 in the original segmentation that did not have any positive segment id within a distance of 2 voxels (maximum distance) to 0 in the adapted segmentation as well. In addition, the ground truth synapse segmentation was modified by keeping only synapse objects that are marked as 'sure'. For interface classification, the bounding box $[2817, 6912; 7041, 10112; 897, 1408]$ was tiled into non-overlapping cubes of size $512 \times 512 \times 256$ voxels. Interface and feature calculation was done for each tile using interface subvolumes with distances of $40\,\mathrm{nm}$, $80\,\mathrm{nm}$ and $160\,\mathrm{nm}$ and Gaussian kernels adapted to the data voxel size of $6 \times 6 \times 30\,\mathrm{nm}^3$. For interface label generation, the pre- and postsynaptic segments of all synapses in the ground truth were annotated in webKnossos (Boergens et al., 2017). The interfaces between all pre- and postsynaptic segments of a synapse with an interface centroid at a maximal distance of $750\,\mathrm{nm}$ to the corresponding synapse centroid that were also overlapping with the corresponding ground truth synapse segmentation object were associated to that synapse and assigned a unique id. All interfaces with a center of mass in the region ac3 with the bounding box $[5472, 6496; 8712, 9736; 1000, 1256]$ were collected and used as the test set for performance evaluation. All interfaces with a center of mass at a distance of at least $2\,\mu\mathrm{m}$ to ac3 were used for training. In addition, interfaces with a center of mass at a distance of at least $1\,\mu\mathrm{m}$ to ac3 were used for training if there was no interface in the test set between the same segment ids.

SynEM was trained with the same hyperparameter as for the SBEM dataset using LogitBoost with 1500 weak learners, a learning rate of 0.1, a cost of 100 for the synaptic class (see subsection 3.3.3) and the directed label set. The evaluation on the test set was done based on all interfaces for a synapse as described in subsection 3.4.1, i.e. a synapse was considered detected (TP) if any interface associated to the synapse was detected. The performance of SynEM and the performance of VesicleCNN as reported in Roncal et al. (2015) are shown in Figure 3.13. SynEM outperforms VesicleCNN, which was developed specifically for this dataset and evaluated on the region ac3 as well. Note that the VesicleCNN output is a voxel probability map and the evaluation in Roncal et al. (2015) was done based on overlap of ground truth objects with predicted objects and not on interfaces between segmented processes.

3.4.5. Application to Connectomes

Next, the test set performance of SynEM was used to estimate neuron-to-neuron errors based on the detection performance for single synapse (see subsection 3.3.4). The recall of excitatory neuron-to-neuron connections by SynEM was estimated using that cortical connections are typ-

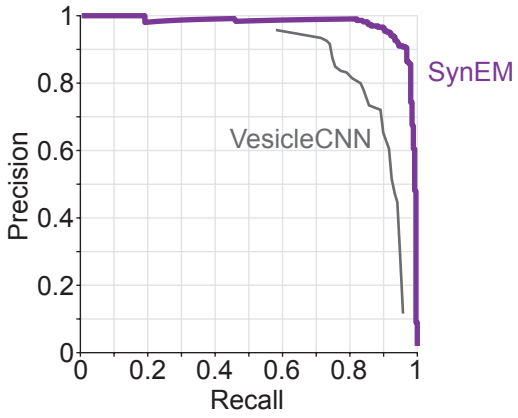

Figure 3.13: Synapse detection performance on an ATUM dataset. SynEM performance (purple) on a dataset from mouse S1 (Kasthuri et al., 2015) and comparison to VesicleCNN (Roncal et al., 2015). (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 3 - Figure supplement 4, legend modified, https://doi.org/10.7554/eLife.26414.014 licensed under CC BY 4.0)

ically established by multiple synapses (Feldmeyer et al., 1999, 2002, 2006; Frick et al., 2008; Markram et al., 1997, range 1 - 8, 4.3 ± 1.4 (mean $\pm$ std.) for excitatory connections; see Table 3.3) thus requiring only a subset of all synapses to be detected. For a binarization threshold of $\gamma_{nn} = 1$, i.e. if a single detected synapse is enough to consider two neurons as connected, a single synapse detection recall R_s of 65.1% results in a neuron-to-neuron recall R_{nn} of 97.1% (Figure 3.14a). The corresponding single synapse precision at this recall was 99.4%. The neuron-to-neuron precision was calculated by randomly distributing false positive synapse detections across all neuron-to-neuron connections and estimating the probability that a previously unconnected pair is assigned a false positive synapse detection. The number of false positive synapse detections was estimated based on the classifier performance as well as the connectivity ratio c_r which is about 20% for local excitatory connections in rodent cortex (Feldmeyer et al., 1999). SynEM achieved a neuron-to-neuron precision P_{nn} of 95.5% at the score threshold corresponding to a neuron-to-neuron recall R_{nn} of 97.1% (Figure 3.14b).

To estimate the neuron-to-neuron error for inhibitory connections, a fixed number of 6 synapses per connected neuron pair (Gupta et al., 2000; Hoffmann et al., 2015; Koelbl et al., 2015; Markram et al., 2004; see also Table 3.4) and a connectivity ratio of $c_r = 0.6$ (Gibson et al., 1999; Koelbl et al., 2015) was used. Since the single synapse detection performance rates used for excitatory connections were calculated on a dense test set with a high fraction of excitatory synapses, a separate inhibitory test set was generated (see subsection 3.4.2). The inhibitory test set consisted of all interfaces between three inhibitory axons and all neighboring processes. As for the dense test set, all interfaces overlapping with a synapse were grouped resulting in a total number of 171 synapses and 9430 non-synaptic interfaces in the inhibitory test set. Performance evaluation was done using precision and recall curves as described for the dense test set requiring that only one of the interfaces overlapping with a synapse needed to be detected to consider the synapse detected. A synapse was considered a FN detection if none of the overlapping interfaces were detected. Predicted synaptic interfaces not overlapping with any ground truth synapse were considered FPs. The best performance on the inhibitory test set was achieved at 75% recall and 82% preci-

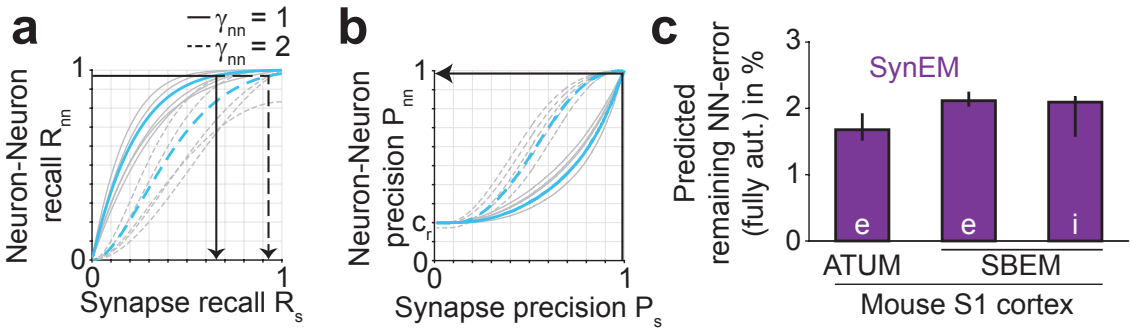

Figure 3.14: Estimated neuron-to-neuron error rates. (a) Estimated neuron-to-neuron recall R_{nn} as a function of the recall R_s for single synapse detection and the number of synapses γ_{nn} to consider a neuron-to-neuron pair connected (binarization threshold, see Figure 3.7a). (b) Estimated neuron-to-neuron precision P_{nn} as a function of the precision P_s for single synapse detection and the binarization threshold γ_{nn}. (c) Predicted remaining neuron-to-neuron error calculated as (1 - F1) score from the SynEM performance on the ATUM-SEM test dataset (created from the data of Kasthuri et al., 2015) and the SBEM dense and inhibitory test datasets using the excitatory and the inhibitory neuron-to-neuron error model, respectively. The connectivity ratios were set to $c_{re} = 20\%$ in the excitatory and $c_{ri} = 60\%$ in the inhibitory model. Error bars, indicated by black lines, specify the range of the error for $c_{re} = 5\%, 10\%, 30\%$ in the excitatory model and $c_{re} = 20\%, 40\%, 80\%$ in the inhibitory model. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 5c-e, `https://doi.org/10.7554/eLife.26414.019` licensed under CC BY 4.0)

Threshold score	**Single synapse** P_s/R_s	**Neuron-to-neuron** P_{nn}/R_{nn}	
		$\gamma_{nn} = 1$	$\gamma_{nn} = 2$
$\theta_s = -1.67$ (exc)	$88.5\%/88.1\%$.	$72.5\%/99.7\%$	$98.1\%/95.6\%$
$\theta_{nn} = -0.08$ (exc)	$99.4\%/65.1\%$.	$98.5\%/97.1\%$	$100\%/83.4\%$
$\theta_s = -2.06$ (inh)	$82.1\%/74.9\%$.	$77.1\%/100\%$	$92.7\%/99.5\%$
$\theta_{nn} = -1.58$ (inh)	$88.6\%/67.8\%$.	$84.7\%/99.9\%$	$97.3\%/98.5\%$

Table 3.7: **SynEM score thresholds.** Overview of the SynEM score thresholds for the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) at the optimal single synapse performance θ_s and the optimal estimated neuron-to-neuron performance θ_{nn} for the excitatory and the inhibitory connectivity model at binarization thresholds of $\gamma_{nn} = 1$ and $\gamma_{nn} = 2$ with the corresponding single synapse precision P_s and single synapse recall R_s and the the estimated neuron-to-neuron precision P_{nn} and recall R_{nn}, respectively. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Table 3 `https://doi.org/10.7554/eLife.26414.028` licensed under CC BY 4.0)

sion (Figure 3.15). The corresponding precision and recall rates of inhibitory neuron-to-neuron connections were estimated at 98% and 97%, respectively.

The remaining neuron-to-neuron error achieved by SynEM was less than 3% (measured as $(1 -$ F1 Score$)$, see Figure 3.14c). The results were stable against variations in the underlying connectivity rates between 5% and 30% for excitatory (c_{re}) and 20% to 80% for inhibitory (c_{ri}) connections. The performance of SynEM is thus likely to be sufficient for most analyses of binary connectomes due to the inherent variations in synaptic connectivity that have been found in biological circuits so far (Helmstaedter et al., 2013). The thresholds for the optimal single synapse performance and the estimated neuron-to-neuron error rates for both the excitatory and the inhibitory connectivity model are summarized in Table 3.7.

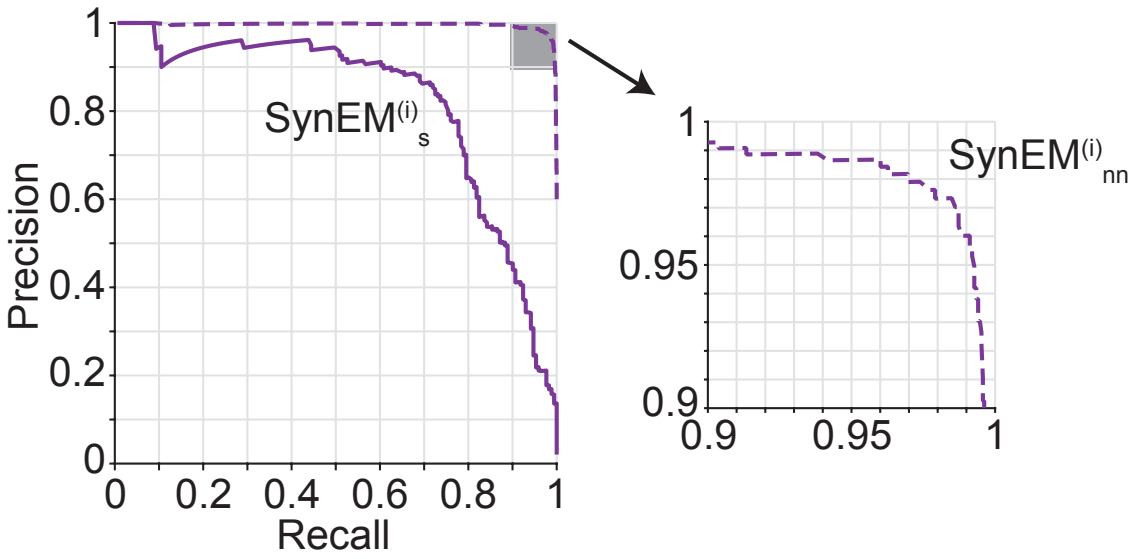

Figure 3.15: Inhibitory synapse detection performance. SynEM performance on a test set of all interfaces along 3 inhibitory axons (171 synapses, 9430 interfaces; dataset 2012-09-28_ex145_07x2 by Boergens and Helmstaedter, 2012b) for single synapses ($\text{SynEM}_s^{(i)}$, solid line) and the estimated neuron-to-neuron errors ($\text{SynEM}_{nn}^{(i)}$, dashed line) assuming 6 synapses for connected neurons. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 5c-e, https://doi.org/10.7554/eLife.26414.034 licensed under CC BY 4.0)

Local Cortical Connectome

To exemplify the application of SynEM for connectomics, a sparse local connectome between 104 axons and 100 postsynaptic processes was reconstructed from the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b; Figure 3.16). The neurites were reconstructed by skeletonization using webKnossos (Boergens et al., 2017) with 1-5 fold redundancy for axons and one-fold redundancy for postsynaptic processes. 10 out of the 104 axons were classified as inhibitory while the remaining 94 were excitatory. The nodes of the skeleton reconstructions were overlapped with the segments of the SegEM segmentation for this dataset (see subsection 3.4.1; Berning et al., 2015) to yield a volume reconstruction for the pre- and postsynaptic processes (Figure 3.16a). For each pair of processes all interfaces were determined and the total border area was calculated using the area calculation proposed in Berning et al. (2015) (Figure 3.16b). Afterwards, each interface was classified by SynEM using the threshold θ_{nn} (Table 3.7) optimized for neuron-to-neuron error resulting in estimated precision and recall rates of 98.5% and 97.1% for excitatory neuron-to-neuron connections and 97.3% and 98.% for inhibitory neuron-to-neuron connections (Figure 3.16c). To account for synapses that were split into multiple interfaces due to the SegEM oversegmentation, the detected synaptic interfaces were clustered if they were closer than $1.5\,\mu\text{m}$ and between the same process pre- and postsynaptic process (Figure 3.17). The weighted connectome was thresholded with a binarization threshold $\gamma_{nn} = 1$ for excitatory and $\gamma_{nn} = 2$ for inhibitory connections (Figure 3.16d). The resulting binary connectome contained a total number of 536 connections with estimated neuron-to-neuron precision and recall rates of 98.5% and 97.1% for excitatory and 97.3% and 98.5% for inhibitory connections, respectively. The total synapse count restricted to the identified connections was 880.

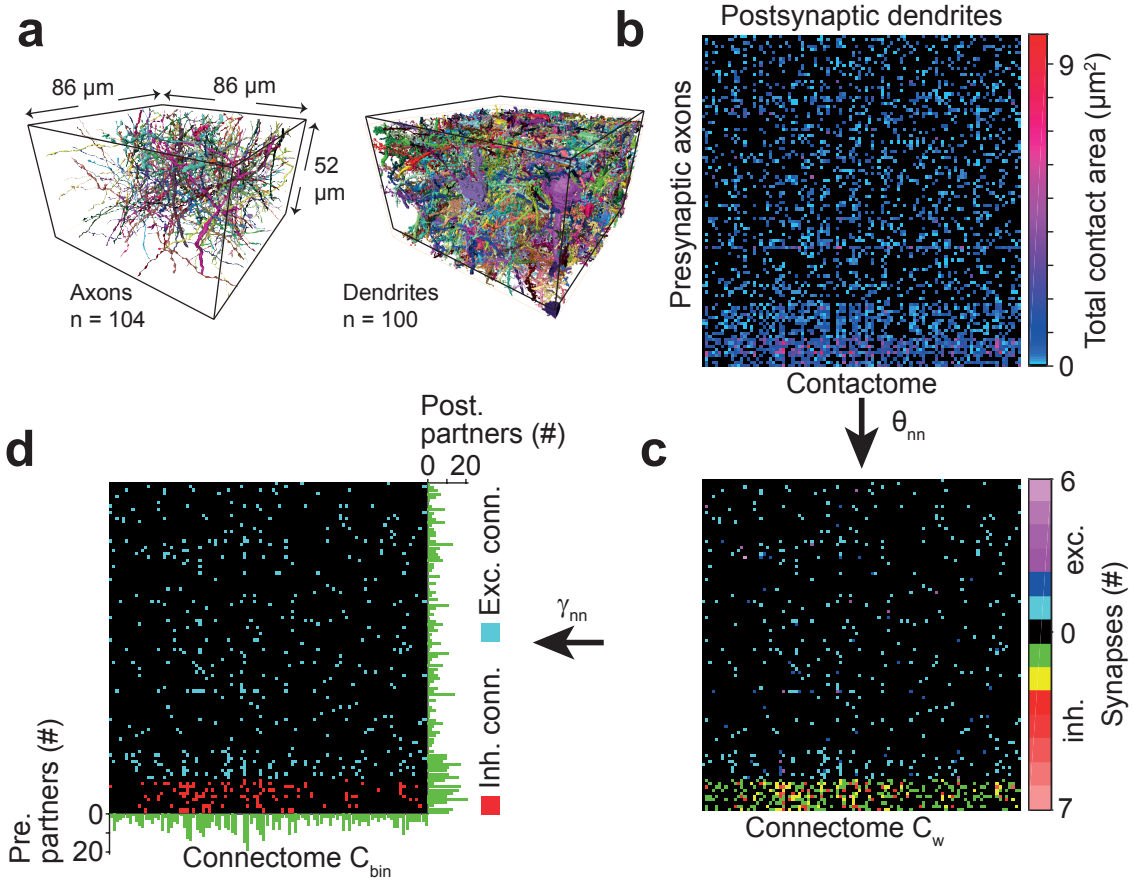

Figure 3.16: Local sparse connectome obtained for a SBEM dataset using SynEM. (a) Volume reconstructions of 104 axonal (presynaptic) and 100 dendritic (postsynaptic) processes from skeleton tracings in a volume of size $86 \times 52 \times 86\,\mu m^3$ of the dataset 2012-09-28_ex145_07x2 from mouse S1 L4 (Boergens and Helmstaedter, 2012b). (b) Contactome reporting the total contact area between pre- and postsynaptic processes. (c) Weighted connectome reporting the total number of synapses at the SynEM threshold θ_{nn} optimized for neuron-to-neuron performance for the respective presynaptic type (excitatory, inhibitory) (see Table 3.7). (d) Binary connectome obtained from the weighted connectome by thresholding at $\gamma_{nn} = 1$ for excitatory and $\gamma_{nn} = 2$ for inhibitory connections resulting in predicted precision and recall rates of 98% and 98% for excitatory and 98% and 97% for inhibitory connections, respectively. (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 6, `https://doi.org/10.7554/eLife.26414.025` licensed under CC BY 4.0)

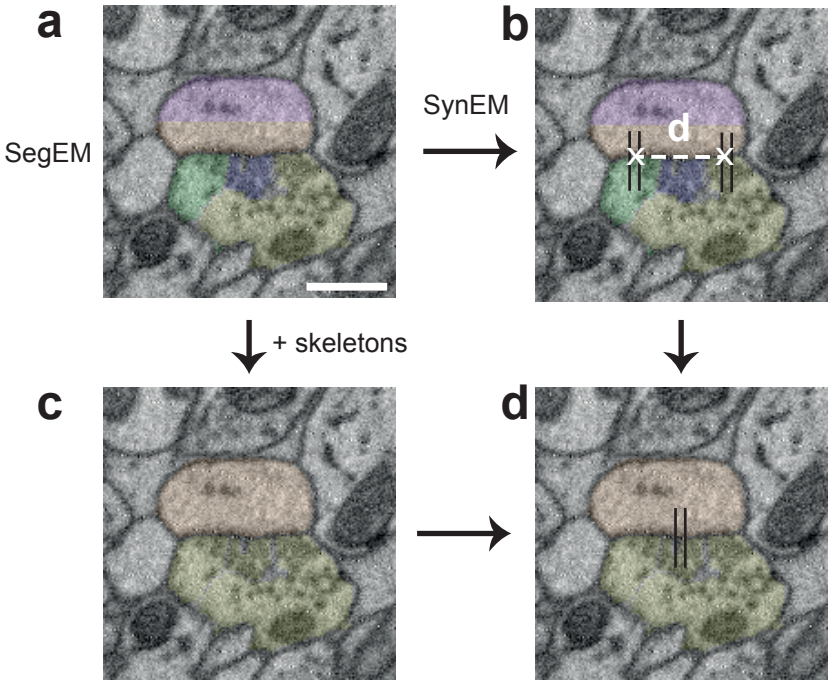

Figure 3.17: Synapse detection for the sparse local connectome. (a) The SegEM segmentation (Berning et al., 2015) of the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) depicted for an axonal (bottom) and dendritic (top) process of the local connectome. Note that an oversegmentation with a bias towards splits was used. (b) Synaptic interfaces (black lines) detected by SynEM between the pre- and postsynaptic process. (c) Skeletons were combined with the segmentation to yield the full neurite volume reconstructions. (d) Synaptic interfaces between the same pre- and postsynaptic process were clustered if they were closer than $d = 1.5\,\mu\text{m}$ (hierarchical clustering) to obtain the synapse counts in the final connectome (Figure 3.16). Scale bar: 500 nm (a-d). (Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, eLife, Figure 6 - Figure supplement 1, `https://doi.org/10.7554/eLife.26414.027` licensed under CC BY 4.0)

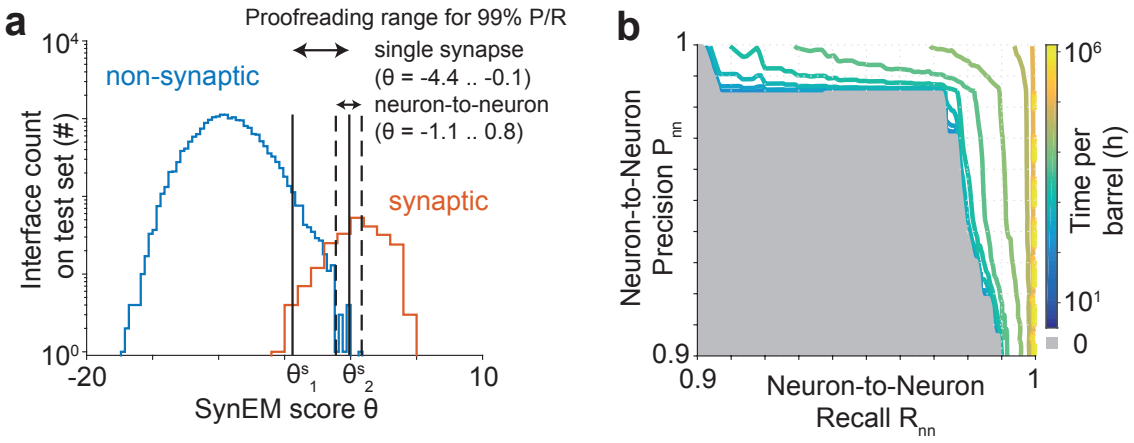

Figure 3.18: Focused proofreading. (a) Score distribution for synaptic and non-synaptic interfaces on the test set. SynEM score ranges for focused proofreading with target precision and recall rates of 99% for single synapse performance (solid, θ_1^s, θ_2^s) and estimated neuron-to-neuron error (dashed). (b) Estimated neuron-to-neuron precision and recall rates and associated focused proofreading cost in hours linearly extrapolated from the test set assuming 10 seconds per interface manual annotation time. The gray area corresponds to the fully automated SynEM performance and can be achieved with 0 proofreading hours. Note that the time colorbar scale is logarithmic. (b is an updated version of Staffler, Berning, Boergens, Gour, van der Smagt, Helmstaedter 2017, biorXiv, Figure 3f, `https://doi.org/10.1101/099994` licensed under CC BY NC ND 4.0)

Focused Proofreading

To improve the accuracy of synapse detection beyond the fully automated result, the SynEM predictions can be used to focus human proofreading. After the specification of the required precision and recall rates for single synapses or neuron-to-neuron connections, it is enough to proofread the SynEM predictions in a limited score range $[\theta_1, \theta_2]$. The lower score θ_1 is determined by the required recall while the upper score θ_2 is determined by the required precision. The lower and upper score thresholds can be estimated from the performance on a test set: The lower threshold θ_1 is chosen as the highest threshold for which the SynEM classifier achieves the desired recall. The upper threshold is defined implicitly as the threshold θ_2 such that after discarding all non-synaptic interfaces in the range $[\theta_1, \theta_2]$ the classifier achieves the desired precision. Alternatively, the upper threshold can be chosen as the lowest threshold for which the SynEM classifier achieves the desired precision potentially resulting in a higher overall precision and thus additional proofreading effort. For SynEM, single synapse precision and recall rates of 99% require to proofread all interfaces in the score range $[-4.4, 0.1]$, while neuron-to-neuron recall and precision rates of 99% result in the score range $[-1.1, 0.8]$ (Figure 3.18a). The annotation cost can be estimated from the test set as well based on the number of interfaces in the determined score range. Thus, for each pair of desired precision and recall rates, the required annotation time for a specified single synapse performance or neuron-to-neuron performance can be estimated and extrapolated to estimate the annotation time for large datasets such as a barrel dataset of size $250 \times 250 \times 250\,\mu\text{m}^3$ (Figure 3.18b).

3.5. Discussion

SynEM, a novel method for the automated detection of chemical synapses in EM-based connectomic datasets was reported. In addition to synapse size and location, SynEM provides the pre- and postsynaptic partners and the synapse direction which is required for connectomic reconstruction. The fully automated performance of SynEM achieves error rates for binary neuron-to-neuron connections below 3% in the mammalian cortex thereby removing synapse detection as the bottleneck in connectomic reconstruction.

In contrast to previous synapse detection methods, SynEM combines the detection of synapses and the partner detection task into a single classification step. This is achieved by classifying interfaces between segments from volume segmentation of the EM data. The formulation of synapse detection as interface classification allows including prior knowledge about possible synapse locations, their orientation and the natural inclusion of the partner detection step. The SynEM feature representation solely relied on local texture and shape features partially based on Kreshuk et al. (2011) and Becker et al. (2013) and did not explicitly include the biological identity of processes such as axons and dendrites or even subcellular classes such as spine, shaft or somatic segments. However, the biological identity of segments could easily be incorporated as additional features if available. Training data annotation for interface classification is also simpler and less time-consuming compared to voxel-based approaches because each synapse only requires the annotation of the pre- and postsynaptic partner at the synapse location instead of voxel labeling (Kreshuk et al., 2011; Becker et al., 2013; Roncal et al., 2015; Dorkenwald et al., 2017) which is often ambiguous for medium resolution data. Furthermore, interface classification offers a simple strategy for focused proofreading of SynEM predictions to improve the accuracy beyond what can be achieved fully automatically. On the other hand, the quality of the volume segmentation affects the SynEM prediction in a non-trivial way. Experiments with local improvements of the segmentation quality at synapse locations indicated that better segmentations also result in higher synapse detection performance (see subsection 3.4.2). However, it is not clear whether the approach of SynEM to consider the whole contact surface for classification is beneficial in all situations, for example when a synapse has a very large contact area but only a small active zone/PSD as is often the case for shaft or soma synapses. In such cases, a more locally confined classification could have advantages.

SynEM does not distinguish between excitatory or inhibitory synapses and the identification of synapse types for the local cortical connectome (subsection 3.4.5) was based on the identity of the presynaptic axons which was determined manually. Traditionally, the identification of synapse types in EM datasets could only be done based on the appearance of the single synapse itself (Gray, 1959; Colonnier, 1968) due to the limitation of dataset sizes to a few slices or even only a single slice containing the synapse. In large-scale 3D EM datasets, however, whole axons can be classified based on multiple synapses along the axon (Kasthuri et al., 2015), including their postsynaptic target structures (shaft/spines/somata) or even on the neuron morphology in case the axon and the corresponding cell body are mostly contained in the dataset. The classification of

synapse types can thus be replaced by the identification of the presynaptic process, which is part of the SynEM classification, and a separate type classification for presynaptic processes. Since different types of synapses can have slightly different textural appearance thus affecting the certainty of the interface classification, the classification can be optimized separately for different classes of pre- and postsynaptic processes. A simple approach was exemplified for the inhibitory test set (subsection 3.4.2), where the same classifier was used for excitatory and inhibitory axons with decision thresholds optimized for the respective synapse types, which is motivated by the fact that inhibitory synapses typically have a less pronounced PSD than excitatory synapses. Alternatively, different synapse types could be learned directly by a multi-class interface classifier or using dedicated classifiers trained only on synapses between specific classes of pre- and postsynaptic processes.

Automated synapse detection methods have first been proposed for high-resolution datasets, for example obtained by FIBSEM with an isotropic resolution of $4\,\mathrm{nm}$–$8\,\mathrm{nm}$, which evidently facilitates synapse detection (Figure 3.11a). However, the generation of large connectomic datasets containing substantial parts of brain circuitry is currently only realistic by compromising resolution for speed (Figure 3.11a). Furthermore, synapse detection methods developed for high-resolution datasets showed a substantial drop in performance when applied to medium-resolution datasets (Figure 3.11). SynEM was explicitly designed to allow for synapse detection in lower resolution datasets that is scalable to large volumes. The superior performance of SynEM on a high-resolution anisotropic ATUM-SEM dataset (Figure 3.13) indicates that segmentation-based interface classification can also improve synapse detection for a wider range of 3D EM methods even beyond the benefit of the integrated partner detection.

While the imaging resolution is limited by the employed EM technique, special fixation procedures that preserve or even enhance extracellular space (ECS) were also reported to aid human annotators with synapse detection (Pallotto et al., 2015). ECS separates most cellular processes making a direct touch between neurons much more indicative of a synaptic contact (chemical or electrical) which should thus facilitate automated synapse detection as well. SynEM was evaluated on conventionally stained and fixated EM data without post-staining or ECS preservation showing its applicability to this widely used data that does not require any special fixation protocols. Changes to the data generation process that facilitates synapse detection for human annotators are thus likely to also improve the performance of SynEM further.

Together, SynEM provides a substantial improvement towards fully automated synapse detection for cortical connectomics in mammals, shifting the analysis bottleneck again towards cellular process reconstruction.

4. Deep Learning for Semantic Segmentation in Connectomics

In this chapter, the application of techniques from deep learning for semantic segmentation tasks in connectomics are described. Semantic segmentation refers to the voxelwise prediction of semantic classes (see also subsection 2.2.3). In section 4.3, the SynEM interface representation is modified with feature maps learned by fully convolutional networks (FCNs) trained on a multi-class classification of synaptic junctions, vesicles and mitochondria. The resulting synapse classification performance is compared to the classifier using the hand-designed feature representation from chapter 3. In section 4.4, FCNs are used for membrane prediction between cellular processes with the goal to improve the volume segmentation underlying the SynEM interface definition. The results were evaluated on the data from the SegEM segmentation challenge (Berning et al., 2015).

4.1. Introduction

Artificial neural networks (ANNs) and in particular deep learning techniques such as convolutional neural networks (CNNs) have enabled remarkable progress in computer vision in recent years in domains such as image classification and object detection (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Szegedy et al., 2015; He et al., 2016), semantic segmentation (Long et al., 2015) and image generation (Rezende et al., 2014; Goodfellow et al., 2014; Gatys et al., 2016). Deep learning has also widely been applied for automated reconstruction in connectomics. CNNs are commonly used to create volume segmentations of cellular processes (Jain et al., 2010b) but also for the segmentation of various ultrastructure such as synaptic junctions and myelin (Dorkenwald et al., 2017). Other use-cases are the automated agglomeration of segments in volume segmentations (Bogovic et al., 2013), the automatic detection of errors in volume segmentations (Zung et al., 2017; Rolnick et al., 2017) and image alignment (Jain, 2017).

The focus of this chapter is on two semantic segmentation tasks with the aim to replace the hand-designed feature representation of SynEM by a learned representation and to improve the underlying volume segmentation used to define interfaces in SynEM. The interface feature representation of SynEM proposed in subsection 3.3.1 is based on hand-designed texture and shape features. However, feature design is a difficult task that requires domain knowledge and the feature representation potentially needs to be adapted to different datasets. Furthermore, it has been shown that learned features are competitive to hand-designed features for the task of agglomeration of volume segmentations by boundary classification in connectomics (Bogovic et al., 2013). In section 4.3, all image filters in the SynEM interface feature representation were replaced by the output channels of FCNs trained to predict synaptic junctions, vesicle clouds and mitochondria, which improved classification performance compared to the hand-designed image filters. Similar to Bogovic et al. (2013), the combination of hand-designed and learned representation resulted in

the best performance.

Taking a step back, the definition of interfaces itself was based on a volume segmentation, and the quality of the segmentation affected the performance of SynEM (subsection 3.4.2), for example when a synapse is split into multiple small interfaces with less context or, obviously, if the synaptic location is not admissible due to a merge error. The generation of volume segmentations is often done in a multi-stage process starting with the calculation of an initial over-segmentation that is subsequently refined by an agglomeration procedure. Two main approaches are currently used to produce the initial over-segmentation based either on boundary or affinity maps that subsequently need to be partitioned into single objects, for example using simple connected components or a watershed procedure (Jain et al., 2010b). Affinity maps indicate whether two voxels belong to the same object (Turaga et al., 2010). Boundary maps specify which voxels separate two processes, for example voxels that belong to membranes between cellular processes, thus containing less information than affinity maps which is, however, often sufficient (Berning et al., 2015). Both boundary maps and affinity graphs have been calculated using a variety of methods, however, it has been shown that in both cases CNNs typically outperform other approaches (Jain et al., 2007; Turaga et al., 2010; Jain et al., 2010b; Ciresan et al., 2012). In section 4.4, FCNs were used to generate boundary maps from electron microscopy (EM) data to create an initial oversegmentation in a segmentation procedure as proposed by SegEM (Berning et al., 2015). Multiple network architectures were compared including regular FCNs built from successive convolutional and pooling operations, a novel multi-resolution architecture that processes features at different resolutions and networks with an encoder-decoder structure (Ronneberger et al., 2015; Funke et al., 2018). The best network showed an improvement of the segmentation result of SegEM (Berning et al., 2015) by a factor larger than two of the inter-error distance (IED). Furthermore, an ensemble of networks gave the overall best result.

4.2. Related Work

The success of deep learning in fields such as image classification has led to a rapid adoption of techniques from deep learning for tasks in semantic segmentation (see for example Long et al., 2015; Chen et al., 2018). The inherent challenge of semantic segmentation is to combine long range context required for semantic classification with local information about the exact location of objects (Long et al., 2015). CNNs are ideally suited for voxelwise prediction due to their ability of processing arbitrarily sized inputs efficiently. In connectomics and in particular for membrane prediction, CNNs have been shown to outperform other methods early on (Jain et al., 2007; Turaga et al., 2010; Jain et al., 2010b; Ciresan et al., 2012). However, methods that used CNNs solely consisting of convolutional layer (Jain et al., 2010a; Turaga et al., 2010; Berning et al., 2015) were typically limited by a small field of view or required large filter sizes to achieve sufficient image context. To increase the available image context, CNNs often contain pooling operations that downsample the spatial resolution thus enlarging the field of view. Ciresan et al. (2012) proposed to train a CNN that classifies the center pixel from the field of view of the network as membrane or non-membrane which is repeated for all pixels of an image. This shift and stitch approach is

computationally inefficient, but it is possible to convert a CNN for classification, which can only process inputs of a fixed size, into a FCN by using what has been called skip kernels (Sermanet et al., 2014), filter rarefaction (Long et al., 2015), dilated or atrous convolutions and was used for example in the ZNN framework (Zlateski et al., 2016). An alternative is the max-pooling fragment approach (Giusti et al., 2013; Masci et al., 2013), which produces several independent outputs at each pooling layer corresponding to all possible offsets of the pooling window that are separately processed by the remaining network layers. Max-pooling fragment was, for example, implemented by the elektroNN[1] framework that was used in SyConn (Dorkenwald et al., 2017) to train classifiers for several kinds of ultrastructure including membrane maps. However, networks with pooling layers compromise a larger field of view with less localized predictions since the filters cannot access information at all scales anymore. To overcome the trade-off between localization and sufficiently large image context, encoder-decoder network architectures have been proposed (Long et al., 2015). In encoder-decoder networks, the semantic information is extracted by the encoder that typically gradually reduces the size of the input image while the decoder recovers the high-resolution spatial information. Encoder-decoder networks form the basis of many current semantic segmentation algorithms and have been combined with multi-resolution methods such as spatial pyramid pooling (He et al., 2014) and atrous convolutions (Chen et al., 2018). An example of such a network is the u-net architecture (Ronneberger et al., 2015), which uses contracting and expanding pathways that are connected by skip connections to combine features from different resolutions. The u-net architecture has been widely adopted in medical image analysis including membrane prediction in connectomics (Lee et al., 2017; Funke et al., 2018; Drozdzal et al., 2018). U-net is fully convolutional although not translation equivariant (cf. Equation 2.8) and used an overlap-tile strategy to combine predictions from adjacent input volumes. Other approaches that process information at multiple scales include the usage of multiple parallel convolutions of different sizes in each layer (Grais et al., 2017) or networks where each feature map has a different dilation ratio (Pelt and Sethian, 2018).

The generation of an initial over-segmentation from boundary maps in connectomics is typically done using simple thresholding followed by connected components (Turaga et al., 2010; Jain et al., 2010b) or a watershed procedure (Berning et al., 2015; Funke et al., 2018) that can be learned as well (Wolf et al., 2017). A recent approach with promising results called flood-filling networks (Januszewski et al., 2017) deviates from the two-step process of boundary map calculation followed by a partitioning procedure and combines them into a single step by iteratively refining a binary mask that constitutes a single cellular process. The object mask creation is then repeated for all cellular processes in a dataset.

To compare volume segmentations, several metrics have been proposed for segmentations in general, but also particularly for the task of neurite reconstruction in connectomics (see also Arganda-Carreras et al., 2015). The most basic metrics are voxelwise errors such as the cross-entropy (Equation 2.18) or sum of squares error (Equation 2.16) that can both be used as a loss functions for CNN training (Ciresan et al., 2012; Berning et al., 2015). Region or segmentation based

[1] `elektronn.org`

metrics compare a proposed segmentation with a ground truth segmentation and can typically be used as loss functions for learning such as the rand error (Unnikrishnan et al., 2007; Turaga et al., 2009), the warping error (Jain et al., 2010a), the variation of information (Arbelaez et al., 2011; Kroeger et al., 2013) and the tolerant edit distance (Funke et al., 2017). Evaluation metrics specific to connectomics typically measure the path length of reconstructed processes by comparing a volume segmentation to ground truth skeletons that serve as a sparse approximation to the underlying cellular processes such as the split-merge metric (Berning et al., 2015) and the expected run length (Januszewski et al., 2017). Note that the tolerant edit distance (Funke et al., 2017) can be used to compare a predicted segmentation to a ground truth skeletonization as well.

The segmentation of ultrastructure other than cellular membranes has received less attention in connectomics so far as discussed in context of synapse detection (see section 3.2). Several approaches for the segmentation of synaptic junctions (Kreshuk et al., 2011; Becker et al., 2013; Kreshuk et al., 2014; Neila et al., 2016) and other ultrastructure such as mitochondria (Perez et al., 2014; Neila et al., 2016) have been proposed that rely on hand-designed features. Dorkenwald et al. (2017) used the elektroNN framework to train CNNs to classify synaptic junctions, vesicle clouds and mitochondria and separate CNNs for myelin, somata and blood-vessels. Furthermore, Dorkenwald et al. (2017) showed that their CNN-based approach is superior to all previously proposed methods relying on hand-designed features and to the 2D CNN of VesicleCNN (Roncal et al., 2015). The most recent approaches used encoder-decoder type FCNs based on u-net (Ronneberger et al., 2015) with more refined training procedures. Heinrich et al. (2018) used a u-net like architecture for synaptic cleft segmentation formulated as a regression task. The network architecture was optimized for an isotropic field of view in the hidden layers of the network. Parag et al. (2018) used a u-net like architectures to predict separate labels for pre- and postsynaptic processes formulated as a regression task, which allowed to estimate synapse orientation in addition to synapse locations from the voxelwise predictions.

4.3. Interface Classification with Learned Texture Filters

In this section, the texture features of the SynEM interface representation were replaced or extended by feature maps learned by CNNs and the result was compared to the performance of the hand-designed feature representation (subsection 3.3.1).

4.3.1. Feature Learning for SynEM

To extend SynEM with learned feature maps, the outputs of CNNs were used as additional texture features in the SynEM interface representation (see subsection 3.3.1). To generate relevant features for synapse detection, the CNNs were first trained for voxelwise classification of the 3D EM data into synaptic junctions, vesicle clouds and mitochondria (Figure 4.1) as in Dorkenwald et al. (2017). For comparison, several CNNs with different architectures (Table 4.1), each with $\tanh$ as hidden activations, a softmax output non-linearity were trained using the elektroNN[2] framework.

[2] `elektronn.org`

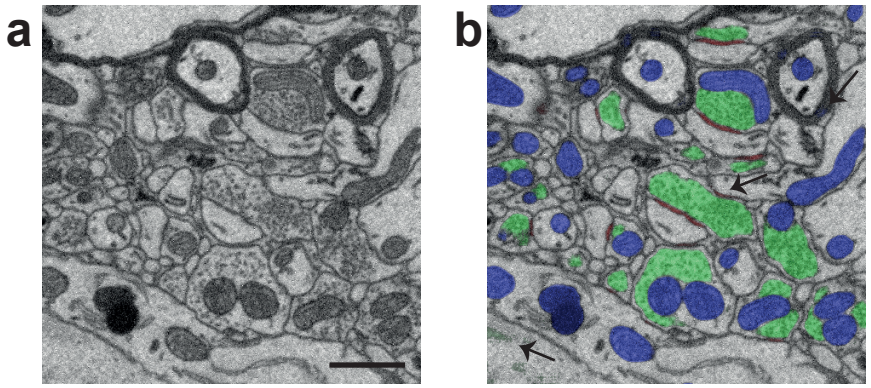

Figure 4.1: CNN ultrastructure prediction in SBEM. (a) Raw data image from the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) from mouse primary somatosensory cortex (S1) layer 4 (L4). (b) Multi-class voxelwise output of a CNN (cnn17_4_large, see also Table 4.1) for synaptic junctions (red), vesicle clouds (green) and mitochondria (blue) overlaid with the raw data from (a). Examples of false positive prediction are marked by arrows for a synaptic junction (center), vesicle cloud (soma in bottom left) and mitochondria (in myelin at top right). Scale bar: 1 µm (a-b).

The multi-class cross-entropy loss (Equation 2.20) was used as loss function with the four classes synaptic junctions, vesicle clouds, mitochondria and a class for background that corresponds to all voxels that do not belong to the first three classes. Each CNN consisted of a sequence of convolutional and max-pooling layers with sizes specified in Table 4.1. ElektroNN ensures full translation equivariance using a volume-based training with sparse outputs and max-pooling fragment for prediction (Giusti et al., 2013; Masci et al., 2013).

For each CNN, three interface feature representations were defined as follows. The first representation used only the raw data and the CNN output channels (synaptic junctions, vesicles, mitochondria, background) as feature maps for texture features and the interface volume and principal axes as shape features and is called "CNN out" in the following. The second representation consisted of the raw data, all feature maps in the last hidden layer and the output layer of the CNN as texture features as well as the interface volume and principal axes as shape features and is called "CNN out & hidden" in the following. The third representation used the CNN output channels in addition to all hand-designed texture features and all shape features from the original SynEM feature representation (subsection 3.3.1) and is called "CNN out & hand" in the following. Each interface classifier is thus uniquely specified by a CNN architecture and the type of feature representation.

4.3.2. Experiments

Classifier Training

Classifier training and evaluation was done on the dataset 2012-09-28_ex145_07x2 from mouse S1 L4 (Boergens and Helmstaedter, 2012b; see also Berning et al., 2015; Staffler et al., 2017a), which was also used for SynEM evaluation (see subsection 3.4.1). For CNN training, the voxel-based training data generated for the comparison of SynEM to SyConn consisting of six training regions of size $3.4 \times 3.4 \times 3.4\,\mu\text{m}^3$ ($300 \times 300 \times 120$ voxels) was used (see subsection 3.4.1) and extended by one additional training region of size $5.8 \times 5.8 \times 7.2\,\mu\text{m}^3$ ($512 \times 512 \times 256$ voxels) annotated in the same fashion. The input data was globally normalized to $[0, 1]$ by dividing it

Name	FOV	Params	Filters sizes	Pooling window sizes	FMs	Lr
cnn17*	[79 79 29]	227740	[8 8 4],[5 5 3],[5 5 3],[3 3 3],[3 3 3], [3 3 3],[3 3 2],[3 3 2],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1]	[12 12 24 24 48 48 48 48 4]	1.00E-03
cnn17_2	[79 79 29]	227740	[8 8 4],[5 5 3],[5 5 3],[3 3 3],[3 3 3], [3 3 3],[3 3 2],[3 3 2],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1]	[12 12 24 24 48 48 48 48 4]	5.00E-04
cnn17_2_re	[79 79 29]	227740	[8 8 4],[5 5 3],[5 5 3],[3 3 3],[3 3 3], [3 3 3],[3 3 2],[3 3 2],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1]	[12 12 24 24 48 48 48 48 4]	1.00E-03
cnn17_2_large	[79 79 29]	322628	[8 8 4],[5 5 3],[5 5 3],[3 3 3],[3 3 3], [3 3 3],[3 3 2],[3 3 2],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1]	[16 16 32 32 48 48 64 64 4]	5.00E-04
cnn17_5	[79 79 29]	227740	[8 8 4],[5 5 3],[5 5 3],[3 3 3],[3 3 3], [3 3 3],[3 3 2],[3 3 2],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1]	[12 12 24 24 48 48 48 48 4]	1.00E-03
cnn17_6	[79 79 29]	227740	[8 8 4],[5 5 3],[5 5 3],[3 3 3],[3 3 3], [3 3 3],[3 3 2],[3 3 2],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1]	[12 12 24 24 48 48 48 48 4]	1.00E-03
cnn17_3	[79 79 29]	227740	[8 8 4],[5 5 3],[5 5 3],[3 3 3],[3 3 3], [3 3 3],[3 3 2],[3 3 2],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1]	[12 12 24 24 48 48 48 48 4]	1.00E-03
cnn17_4_large	[79 79 29]	322628	[8 8 4],[5 5 3],[5 5 3],[3 3 3],[3 3 3], [3 3 3],[3 3 2],[3 3 2],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1]	[16 16 32 32 48 48 64 64 4]	1.00E-03
cnnCD_1	[57 57 23]	249646	[5 5 3],[5 5 3],[5 5 3],[5 5 3],[5 5 3], [5 5 3],[5 5 3],[5 5 3],[5 5 2],[5 5 2], [5 5 2],[5 5 2],[5 5 2],[5 5 2],[1 1 1]	[1 1 1],[1 1 1],[1 1 1],[1 1 1],[1 1 1], [1 1 1],[1 1 1],[1 1 1],[1 1 1],[1 1 1], [1 1 1],[1 1 1],[1 1 1],[1 1 1],[1 1 1]	[12 12 12 12 12 18 18 18 18 18 24 24 24 24 4]	1.00E-4, 5.00E-5
cnnCD_2	[57 57 23]	249646	[5 5 3],[5 5 3],[5 5 3],[5 5 3],[5 5 3], [5 5 3],[5 5 3],[5 5 3],[5 5 2],[5 5 2], [5 5 2],[5 5 2],[5 5 2],[5 5 2],[1 1 1]	[1 1 1],[1 1 1],[1 1 1],[1 1 1],[1 1 1], [1 1 1],[1 1 1],[1 1 1],[1 1 1],[1 1 1], [1 1 1],[1 1 1],[1 1 1],[1 1 1],[1 1 1]	[12 12 12 12 12 18 18 18 18 18 24 24 24 24 4]	1.00E-4, 5.00E-5
cnnDK_1	[111 111 13]	189724	[6 6 1],[4 4 1],[4 4 4],[4 4 2],[4 4 2], [1 1 1]	[2 2 1],[2 2 1],[2 2 2],[2 2 2],[1 1 1], [1 1 1]	[12 24 36 48 48	1.00E-3, 5.00E-4
cnnDK_2	[111 111 13]	189724	[6 6 1],[4 4 1],[4 4 4],[4 4 2],[4 4 2], [1 1 1]	[2 2 1],[2 2 1],[2 2 2],[2 2 2],[1 1 1], [1 1 1]		1.00E-3, 5.00E-4
svm_1	[47 47 15]	204580	[12 12 6],[7 7 3],[5 5 3],[1 1 1]	[2 2 1],[2 2 2],[2 2 2],[1 1 1]	[16 32 48 4]	1.00E-3, 5.00E-4
svm_2a	[63 63 25]	245748	[8 8 2],[5 5 2],[3 3 2],[3 3 2],[3 3 3], [3 3 3],[1 1 1]	[1 1 1],[2 2 1],[2 2 2],[2 2 2],[1 1 1], [1 1 1],[1 1 1]	[16 16 32 48 64 64 4]	1.00E-3, 5.00E-4
svm_2b	[87 87 37]	245748	[8 8 2],[5 5 2],[3 3 2],[3 3 2],[3 3 3], [3 3 3],[1 1 1]	[1 1 1],[2 2 1],[2 2 2],[2 2 2],[2 2 2], [1 1 1],[1 1 1]	[16 16 32 48 64 64 4]	1.00E-3, 5.00E-4
svm_2c	[103 103 41]	246772	[8 8 3],[5 5 2],[3 3 2],[3 3 2],[3 3 3], [3 3 3],[1 1 1]	[2 2 1],[2 2 2],[2 2 2],[1 1 1],[2 2 2], [1 1 1],[1 1 1]	[16 16 32 48 64 64 4]	1.00E-3, 7.50E-4
svm_3a	[69 69 27]	237892	[8 8 2],[3 3 2],[3 3 2],[3 3 2],[3 3 2], [3 3 2],[3 3 3],[3 3 3],[3 3 3],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[1 1 1],[2 2 2], [1 1 1],[1 1 1],[2 2 2],[1 1 1],[1 1 1]	[12 12 24 24 24 48 48 48 48 4]	1.00E-3, 5.00E-4
svm_3b	[81 81 33]	238660	[8 8 3],[3 3 2],[3 3 2],[3 3 2],[3 3 2], [3 3 2],[3 3 3],[3 3 3],[3 3 3],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [1 1 1],[2 2 2],[1 1 1],[1 1 1],[1 1 1]	[12 12 24 24 24 48 48 48 48 4]	5.00E-4, 3.70E-4
svm_3c	[113 113 49]	230884	[8 8 3],[3 3 2],[3 3 2],[3 3 2],[3 3 2], [3 3 2],[3 3 3],[3 3 3],[3 3 3],[1 1 1]	[1 1 1],[2 2 1],[1 1 1],[2 2 2],[1 1 1], [2 2 2],[1 1 1],[2 2 2],[1 1 1],[1 1 1]	[12 12 24 24 36 36 48 48 48 4]	1.00E-3, 7.50E-4
svm_4a	[75 75 31]	374080	[8 8 2],[3 3 1],[3 3 2],[3 3 2],[3 3 2], [3 3 2],[3 3 3],[3 3 3],[3 3 3],[3 3 3], [3 3 3],[3 3 3],[1 1 1]	[1 1 1],[1 1 1],[2 2 1],[1 1 1],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[1 1 1],[1 1 1], [1 1 1],[1 1 1],[1 1 1]	[12 12 12 24 24 24 48 48 48 48 48 48 48 4]	1.00E-4, 7.50E-5
svm_4b	[103 103 45]	374080	[8 8 2],[3 3 1],[3 3 2],[3 3 2],[3 3 2], [3 3 2],[3 3 3],[3 3 3],[3 3 3],[3 3 3], [3 3 3],[3 3 3],[1 1 1]	[1 1 1],[1 1 1],[2 2 1],[1 1 1],[1 1 1], [2 2 2],[1 1 1],[1 1 1],[2 2 2],[1 1 1], [1 1 1],[1 1 1],[1 1 1]	[12 12 12 24 24 24 48 48 48 48 48 48 48 4]	1.00E-4, 7.50E-5
svm_5a	[79 79 33]	329512	[8 8 3],[3 3 2],[3 3 2],[3 3 2],[3 3 2], [3 3 2],[3 3 3],[3 3 3],[3 3 2],[3 3 2], [3 3 3],[3 3 3],[3 3 2],[3 3 3],[3 3 3], [1 1 1]	[1 1 1],[1 1 1],[1 1 1],[1 1 1],[2 2 1], [1 1 1],[1 1 1],[1 1 1],[1 1 1],[2 2 2], [1 1 1],[1 1 1],[1 1 1],[1 1 1],[1 1 1], [1 1 1]	[12 12 12 12 12 24 24 24 24 24 48 48 48 48 48 4]	1.00E-4, 7.50E-5
svm_5b	[117 117 47]	172552	[8 8 3],[3 3 2],[3 3 2],[3 3 2],[3 3 2], [3 3 2],[3 3 3],[3 3 3],[3 3 2],[3 3 2], [3 3 3],[3 3 3],[3 3 2],[3 3 3],[3 3 3], [1 1 1]	[1 1 1],[1 1 1],[1 1 1],[2 2 1],[1 1 1], [1 1 1],[1 1 1],[2 2 2],[1 1 1],[1 1 1], [1 1 1],[2 2 2],[1 1 1],[1 1 1],[1 1 1], [1 1 1]	[12 12 12 12 18 18 18 18 24 24 24 24 36 36 36 4]	1.00E-4, 7.50E-5
svm_segEM	[51 51 21]	248144	[11 11 5],[11 11 5],[11 11 5],[11 11 5],[11 11 5],[1 1 1]	[1 1 1],[1 1 1],[1 1 1],[1 1 1],[1 1 1], [1 1 1]		1.00E-4, 3.00E-5

Table 4.1: **CNNs trained on synaptic junctions, vesicle clouds and mitochondria.** Name: Internal name used for the architecture; FOV: Field of view of the CNN in voxels for the in x, y, z dimension, respectively; Params: Total number of trainable parameters; Filters sizes: Size of the convolutional filter in voxels for the respective dimension and layer; Pooling window sizes: Max-pooling window size in voxels for the respective dimension and layer which is applied after the convolutional filter (pooling operations with a size of 1 in all dimensions correspond to the identity transformation and are not executed in the actual implementation). FMs: Number of output feature maps of the corresponding convolutional layer. Lr: Initial learning rate of the CNN. (*) cnn17 was only trained on the data used for comparison of SynEM with SyConn

name	FM	F1 val	F1 test	AUC val	AUC test
cnn17_4_large	out & hand	0.914	0.892	0.961	0.952
cnn17	out	0.898	0.898	0.950	0.949
svm_2a	out & hidden	0.897	0.890	0.951	0.943
SynEM_tr2	hand	0.865	0.874	0.932	0.93
SynEM	hand	0.865	0.858	0.926	0.921

Table 4.2: **SynEM with learned features validation set performance.** Performance of best classifiers on the validation set ranked by F1 score specified using their feature representation and the underlying CNN architecture. For each type of feature map, the best classifier on the validation set is specified. SynEM refers to the original SynEM classifier trained on the training set from subsection 3.4.1 and on the extended interface training set (SynEM_tr2). name: Internal name used for the CNN architecture as specified in Table 4.1. FM: Type of feature map using only the CNN output layer (out), the CNN output and last hidden layer (out & hidden) and the CNN output and hand-designed feaures (out & hand).

by 255 and data augmentation was done using random rotations (in x-y dimension) and flips as well as histogram-based data augmentation as provided by elektroNN. Each CNN was trained for four days using stochastic gradient descent with an initial learning rate as specified in Table 4.1 (multiple learning rates correspond to successive runs of four days each), a momentum of 0.9 and a learning rate decay of 0.997 every 1000 steps. Interface classifier training was done using the same classifier setup as for SynEM (see subsection 3.3.3) consisting of an ensemble of boosted decision stumps trained using LogitBoost (Friedman et al., 2000), 1500 weak learners, a learning rate of 0.1 and a cost of 100 for the synaptic class. The SynEM interface training data was used and extended by two additional cubes of size $5.8 \times 5.8 \times 7.2\,\mu\mathrm{m}^3$ ($512 \times 512 \times 256$ voxels) each from a newly aligned version of the dataset 2012-09-28_ex145_07x2 (2012-09-28_ex145_07x2_-ROI2017, Boergens and Helmstaedter, 2012c; size $96 \times 64 \times 96\,\mu\mathrm{m}^3$) with a volume segmentation generated by SegEM using the same segmentation parameters as for the SynEM test set (see subsection 3.4.1) resulting in a total number of 2066 synaptic and 84974 non-synaptic interfaces in the training set. For comparison, the SynEM classifier was also retrained using the additional interface training data.

Classifier Evaluation

All interface classifiers were evaluated on the validation set of SynEM (Figure 4.2). The best performance on the validation set according to both F1 score and area under curve (AUC) score of the precision-recall curve were attained by classifiers using a combination of CNN features and hand-designed features ("CNN out & hand") outperforming classifiers that use only the CNN output layer ("CNN out") or the CNN output and last hidden layer ("CNN out & hidden"). The best classifiers with respect to the three different types of feature representations are listed in Table 4.2.

The best classifiers from the validation set ranked by F1 score for each feature representation

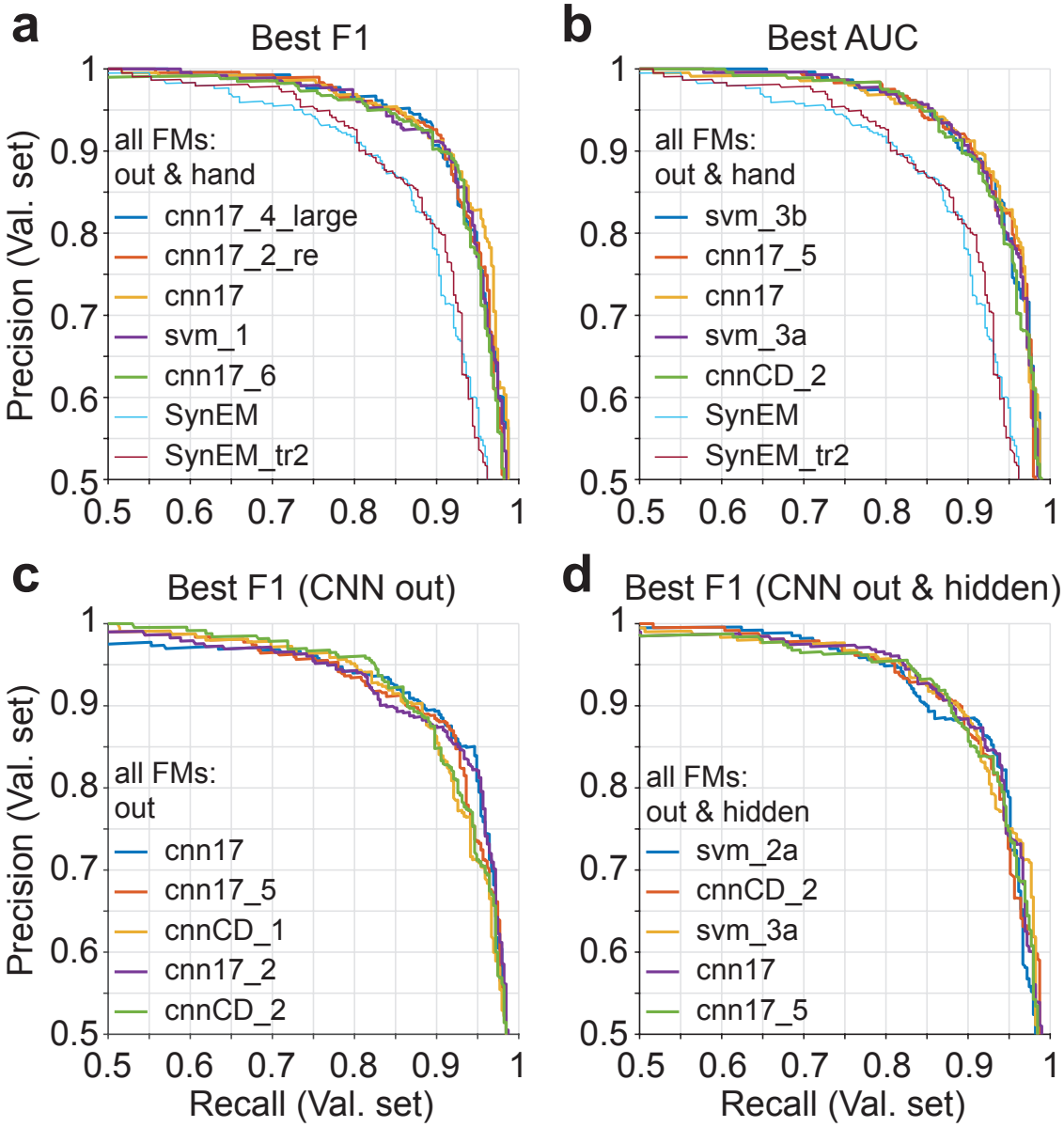

Figure 4.2: SynEM with learned features validation set performance. (a) The five best classifiers (specified using the feature representation and the underlying CNN) ranked by F1 score on the validation set. SynEM refers to the SynEM classifier using only hand-designed features trained on the SynEM training data and extended training data (SynEM_tr2), respectively. Note that all of the classifiers in (a) and (b) used the feature map "out & hand" consisting of a combination of CNN and hand-designed features. (b) The five best classifiers ranked by AUC score of the precision-recall curve on the validation set. (c) The five best classifiers using only the CNN output (CNN out) as texture features ranked by F1 score on the validation set. (d) The five best classifiers using the CNN output and hidden layer as texture features (CNN out & hidden) ranked by F1 score on the validation set.

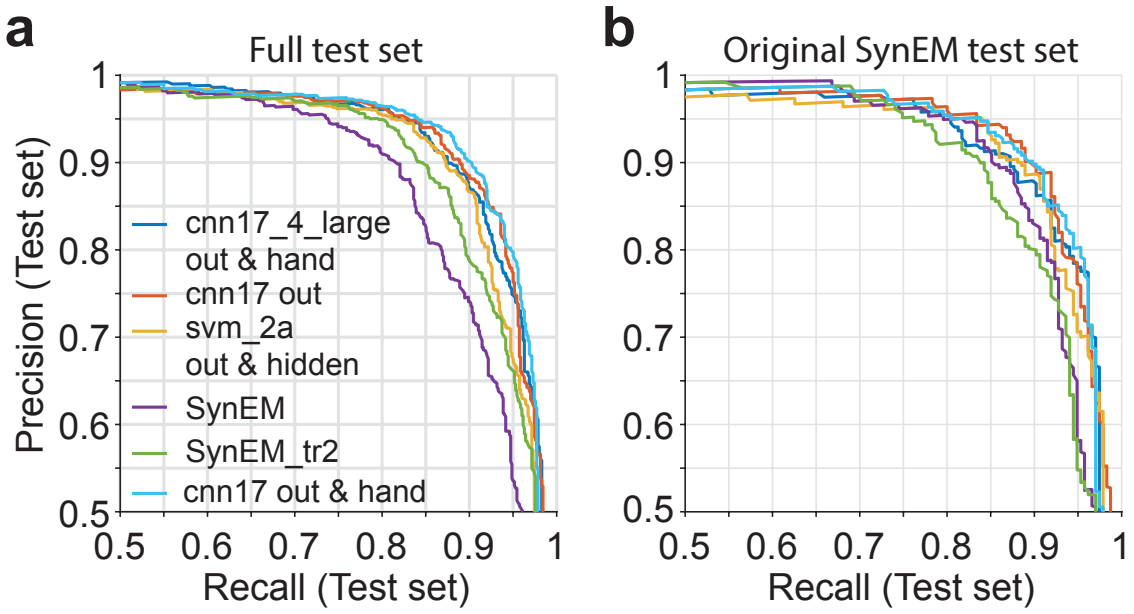

Figure 4.3: SynEM with learned features test set performance. (a) Test set performance of the best classifiers from the validation set for the different feature maps, as well as for SynEM and SynEM with extended training data (SynEM_tr2). Note that the classifier with best validation score (cnn17_4_large) is worse than the best classifier using the feature map based on the CNN output (cnn17 out), which in turn is worse than the corresponding CNN output combined with the hand-designed features (cnn17 out & hand). (b) Test set performance restricted to the test set volume that was used for the evaluation of SynEM in subsection 3.4.2. Note that the original SynEM classifier performs better than the retrained version using the extended training data (SynEM_tr2), which is not true on the larger test set in (a).

were evaluated on a test set of dense synapse annotations within three regions distinct from the training and validation regions (Figure 4.3). The test set consisted of the SynEM test set (subsection 3.4.1) and two additional volumes of size $5.8 \times 5.8 \times 7.2\,\mu\mathrm{m}^3$ ($512 \times 512 \times 256$ voxels) from the newly aligned version of the dataset 2012-09-28_ex145_07x2 (2012-09-28_ex145_07x2_ROI2017, Boergens and Helmstaedter, 2012c) described above. Annotation of the additional test regions was done by the first expert annotator of the SynEM test set by first going through all predictions of SynEM at a recall of 95% and a subsequent search for synapses at missed boutons. For each synapse, all segments in the pre- and postsynaptic process at the synapse location were annotated and all interfaces between any pre- and postsynaptic segment were associated with the corresponding synapse. Interfaces with a center of mass closer than $160\,\mathrm{nm}$ to the border were discarded. If a synapse contained at least one interface with a center of mass further than $160\,\mathrm{nm}$ to the boundary, then all interfaces of the corresponding synapses were kept. One identified ground truth synapse did not overlap with any interface larger than 150 voxels and is thus always counted as a missed synapse (FN). The resulting test set comprised a total number of 532 synapses. Performance evaluation was done as for SynEM requiring that at least one of the interfaces overlapping with a synapse needs to be detected to consider the synapse detected (see subsection 3.4.2). The precision-recall curves on the test set show that all CNN-based classifiers outperform the purely hand-designed approach (Figure 4.3a). In contrast to the validation set, the classifier using only the CNN output (cnn17 out) performed best on the test set. Note that the interface classifier using the cnn17 output and hand-designed features (cnn17 out & hand in Figure 4.3) performed even better than the classifier using only the cnn17 output but has a lower validation set F1 score than the classifier cnn17_4_large out & hand (rank 3 in Figure 4.2a). For

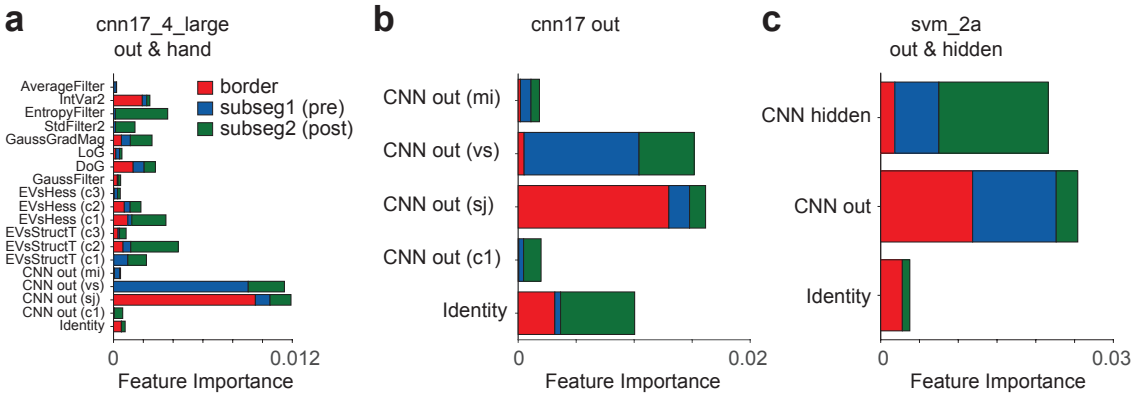

Figure 4.4: Feature importance for interface classification with learned features. Feature importance of texture features for the best interface classifiers on the validation set using the three different types of feature maps with learned features. Feature channels are indicated in brackets with sj: synaptic junctions, vs: vesicles and mi: mitochondria or c1 to c4 for unspecific channels in (a) and (b). (a) Best classifier using the CNN (cnn17_4_large) output and all hand-designed features of SynEM. (b) Best classifier using only the CNN (cnn17) output channels. (c) Best classifier using the CNN (svm_2a) output channels and the last hidden layer. Color code in (b) and (c) is the same as (a).

a better comparison to the results of the SynEM classifier (subsection 3.4.2), the results on the original SynEM test set are shown in Figure 4.3b.

The feature importance of the texture features and subvolumes of the classifiers evaluated on the test set is shown in Figure 4.4. For all three classifiers, the CNN output for synaptic junctions at the interface border volume as well as vesicles in the interface presynaptic subvolume had a high importance as expected, whereas the mitochondria output channel had a comparably low importance. Note that the raw data (identity feature) contributed to the border subvolume for all classifiers despite the explicit CNN output for synaptic junctions. The most important features for the classifier using both CNN and hand-designed features (cnn17_4_large out & hand) were the CNN outputs for synaptic junctions and vesicles but many hand-designed filters also contributed to the classification, in particular for the postsynaptic subvolume (Figure 4.4a). Similarly, for the classifier using only the CNN output (cnn17 out) the raw data had a large contribution, in particular for the postsynaptic volume (Figure 4.4b). The classifier using the CNN output as well as the last hidden layer (svm_2a out & hidden) also used mainly the output layer, however, the hidden layers again largely contribute to the classification, in particular for the postsynaptic subvolume.

Examples of classification errors on the test set for SynEM and for the classifier cnn17_4_large (out & hand) are shown in Figure 4.5. As expected, the additional information contained in the CNN channels was beneficial to avoid false positive detections at mitochondria (Figure 4.5a, first row) but also near myelin and in somata as well as in large dendrites. In addition to the high-level semantic output of the CNN, which was preferentially used by the interface classifier even when other texture features were available (see Figure 4.4), the voxelwise CNN prediction was independent of the segmentation and had a large field of view compared to the hand-designed texture features. This can be beneficial to detect synapses that are split into several interfaces each covering only a small part of the synapse thus providing only limited context for the hand-designed

texture features, which was frequently the case for SynEM FN detections (Figure 4.5b). Similar to SynEM, FP detection of the CNN-based classifier often occurred at incidental non-synaptic touches of a bouton with a neighboring process (Figure 4.5c), however, these FP detections often were at difficult locations that also showed other synaptic features such as a thickening of the corresponding membrane e.g. due to an oblique orientation of the membrane with respect to the imaging plane. FN detections often occurred at small synapses with only a few vesicles and weak indication of a postsynaptic density (PSD) (Figure 4.5d).

The best classifier from the validation set ranked by F1 score (cnn17_4_large out & hand) as well as the best classifier using only the CNN output (cnn17 out) were in addition evaluated on the SynEM inhibitory test set (Figure 4.6a), which consisted of 171 inhibitory synapses along 3 inhibitory axons (see subsection 3.4.1). Despite the improved performance on the dense test set, the CNN based interface classifiers did not show an overall improvement on the inhibitory test set but only for high recall values larger than 80% for which, however, the precision is already below 70%.

The distribution of synapse scores over the whole dataset 2012-09-28_ex145_07x2 is shown in Figure 4.6b. In comparison to SynEM, the CNN-based classifiers show a stronger bimodality with more interfaces that either have very low or very high scores, which is most distinctive for the classifier using only CNN texture features (cnn17 out). Interestingly, the SynEM classifier produces a set of outliers with very high scores corresponding to FP detections in somata and blood vessels, which were not contained in the original SynEM training data. Due to the inclusion of somata in the extended interface classifier training data, the somata FP detections mostly vanished for the SynEM_tr2 classifier, however, the blood vessel FPs were still present. The score distributions of the CNN-based classifier on the other hand do not show systematic FP detections with very high scores despite the fact that also the voxel-based training data did not contain blood vessels.

Generalization to Novel Datasets

To assess the generalization capabilities of interface classifiers, the serial block-face electron microscopy (SBEM) dataset 2017-11-16_ex144_st08x2 (Boergens and Helmstaedter, 2012a; abbreviated with ex144 in the following) from mouse S1 layer 2/3 (L2/3) of size $97 \times 61 \times 196\,\mu\mathrm{m}^3$ with a voxel size of $12 \times 12 \times 26\,\mathrm{nm}^3$ was used. The raw data was segmented using SegEM (Berning et al., 2015). In brief, for membrane prediction the raw data was normalized using the standard score using a global mean of 166.8 and a global standard deviation of 21.0 and the watershed segmentation was run with parameters $r_{se} = 0$, $\theta_{ms} = 10$, $\theta_{hm} = 0.25$.

For testing, a volume of size $6.14 \times 6.14 \times 6.66\,\mu\mathrm{m}^3$ ($512 \times 512 \times 256$ voxels; bounding box $[4737, 5377, 3457, 5248, 5888, 3712]$) was randomly selected. Test set annotation was done by the first expert annotator of the SynEM test set as described in the previous section using the predictions of several classifiers (SynEM, SynEM_tr2, cnn17_4_large out & hand, cnn17 out

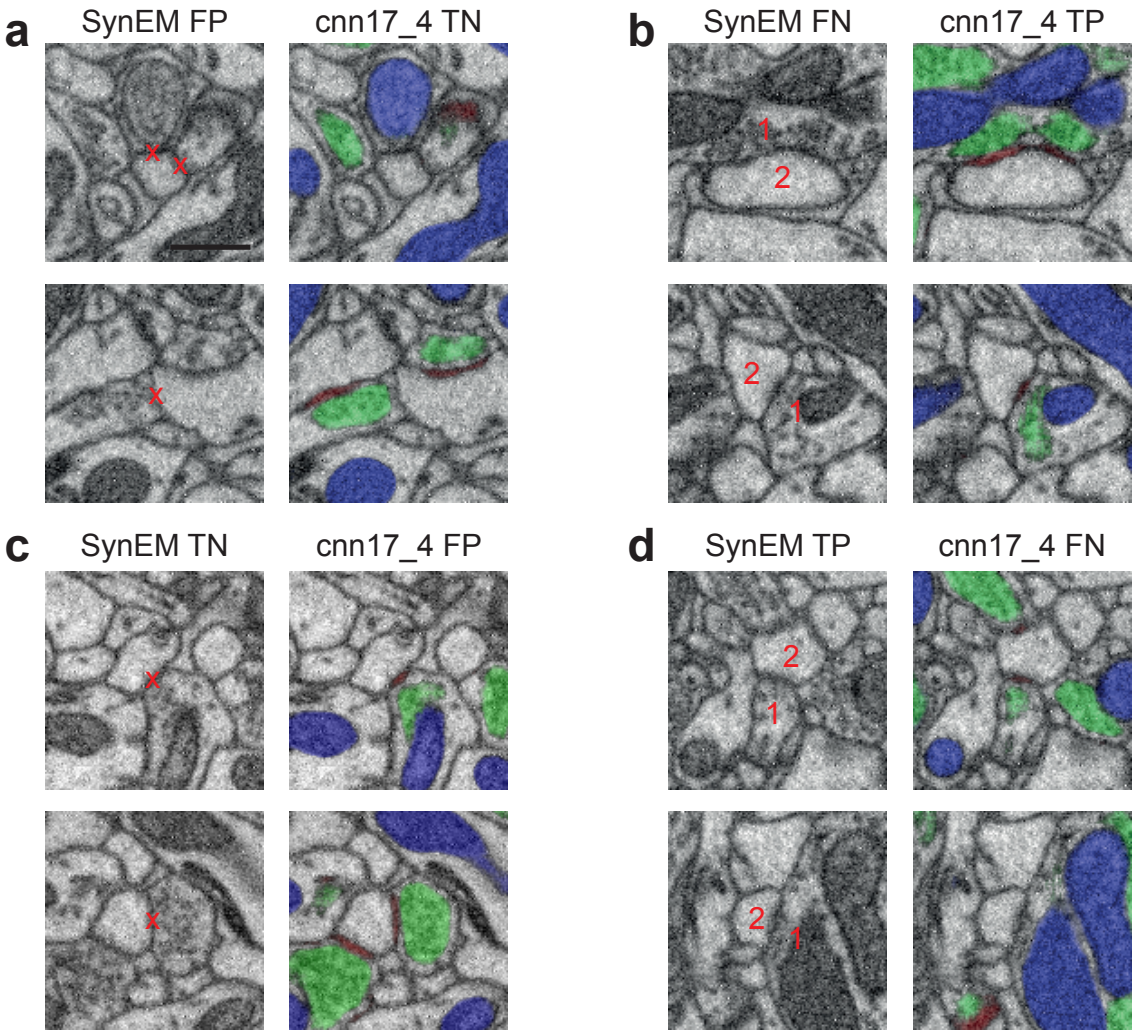

Figure 4.5: Qualitative classification error comparison. Comparison of errors on the test set made by the classifiers SynEM and cnn17_4_large (out & hand) on the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b). In each panel, the raw data (left column) and the corresponding CNN output channels for synaptic junctions (red), vesicle clouds (green) and mitochondria (blue) overlaid with the raw data (right column) are shown. The processes of interest are marked in the raw data images only with 1 and 2 for pre- and postsynaptic, respectively, while x marks interfaces between processes that are non-synaptic. Note that a synapse can consist of multiple interface which is not shown here. (a) False positive (FP) detections of SynEM that were correctly classified as non-synaptic by the cnn17_4_large classifier (TN). (b) False negative (FN) detections of SynEM that were correctly classified as synaptic by the cnn17_4_large classifier (TP). (c) False positive (FP) detections of the cnn17_4_large classifier that were correctly classified as non-synaptic by SynEM (TN). (d) False negative (FN) detections of the cnn17_4_large classifier that were correctly classified as synaptic by SynEM (TP). Scale bar: 500 nm (a). Scale bar in (a) applies to all images in (a-d).

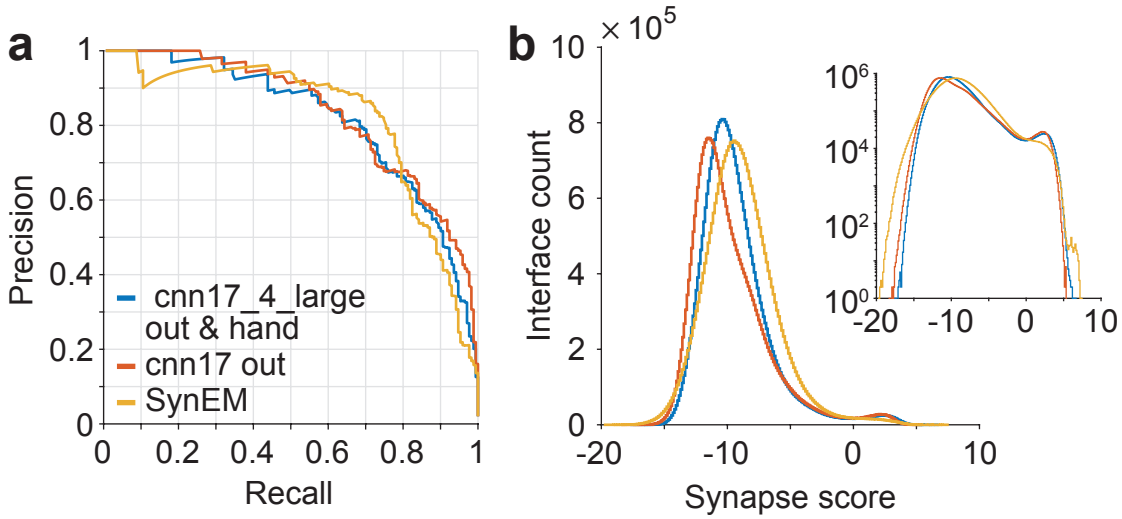

Figure 4.6: SynEM with learned features inhibitory test set performance. (a) Precision-recall curve for the classifier with best overall performance (cnn17_4_large out & hand), the best classifier using only the CNN output features (cnn17 out) and the original SynEM classifier using the hand-designed feature representation on the SynEM inhibitory test set. (b) Classification score histogram for the classifiers in (a) on the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) from mouse S1 L4 for all interfaces with a size above 150 voxels. Inset, score histogram with counts on a logarithmic scale. The large number SynEM outliers with very high scores correspond to FP detections in somata and blood vessels which were not represented in the SynEM training data. Note that the voxel-based training data for CNN training and the extended synapse detection training data did contain parts of somata and the corresponding classifiers do not show systematic FP detections with high scores.

at the threshold corresponding to a recall of 95% on the test set of 2012-09-28_ex145_07x2) in combination with a search for synapses at missed boutons. A second expert neuroscientist selectively proofread the annotations of the first annotator as well as all synapses marked as unsure by the first annotator. The second expert revoked 20 synapse annotations marked as unsure by the first expert and none of the certain synapses. Interface labeling and evaluation was done as described in the previous section. The resulting total number of synapses in the test set was 246. For performance evaluation, it was required that at least one of the interfaces of a synapse was detected to consider the synapse detected (see subsection 3.4.2).

Performance evaluation on the test set from the dataset ex144 was done for several classifiers from the previous section, which were trained only on the dataset 2012-09-28_ex145_07x2 (abbreviated with ex145 in the following) but not on the dataset ex144. The SynEM classifiers (SynEM, SynEM_tr2), the best classifiers from the validation set from the dataset ex145 using the CNN output features (cnn17 out) and the combined feature representation of CNN and hand-designed features (cnn17_4_large out & hand) as well as the classifier with highest F1 score from the test set of the dataset ex145 (svm_3a out & hand) were used. For feature calculation, the raw data of the dataset ex144 was scaled to match the global mean and the global standard deviation of the dataset ex144 (unnormalized mean: 122; unnormalized SD: 22; Berning et al., 2015). The raw data transformation was done by first calculating the standard score for each voxel using the global mean and standard deviation of the dataset ex144 and then the inverse standard score using the global mean and standard deviation of the dataset ex145. Feature calculation was done without any adaption of the features of the classifiers trained on the dataset ex145. The resulting precision-recall curves are shown in Figure 4.7a. For each classifier, the comparison of the opti-

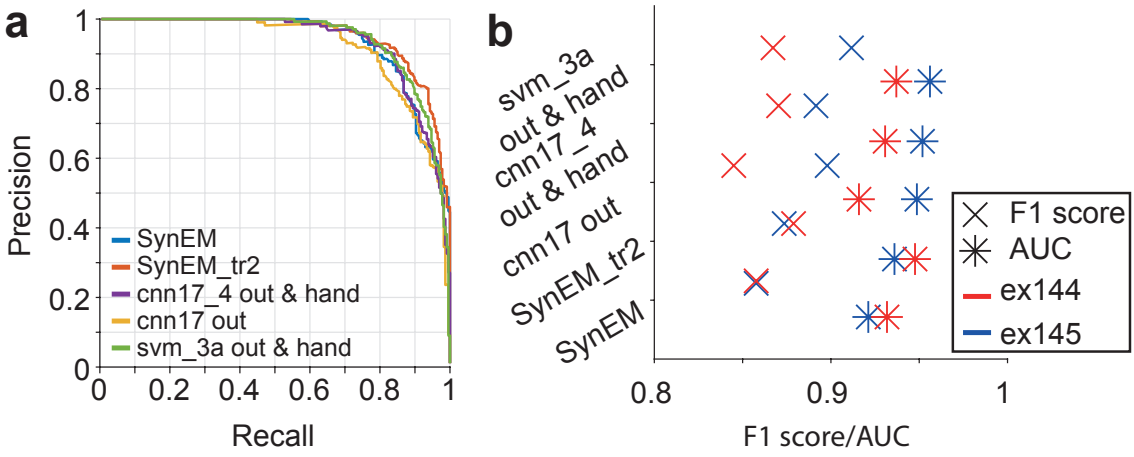

Figure 4.7: Interface classifier generalization performance. (a) Precision and recall curves on the test set of 2017-11-16_ex144_st08x2 (Boergens and Helmstaedter, 2012a) for several classifiers trained on the dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b). (b) Comparison of F1 score (cross) and AUC (asterisk) for the dataset 2017-11-16_ex144_st08x2 (ex144, red) and the test set of 2012-09-28_ex145_07x2 (ex145, blue).

mal F1 scores and AUC scores of the precision recall curves for the test set from 1x145 and the test set from the dataset ex144 are shown in Figure 4.7b.

4.3.3. Discussion

Fully convolutional networks (FCNs) were used to learn texture features directly from the raw data for the SynEM interface classification approach for synapse detection. A feature representation based only on learned texture features resulted in a performance improvement of more than 2% in F1-score. A combination of learned and hand-designed features performed even better on the validation set but was slightly inferior to the fully learned texture feature representation on the test set. Typically, learned features in combination with hand-designed features outperformed the purely learned features (see for example cnn17 in Figure 4.3). In addition, the generalization performance to novel SBEM dataset from mouse S1 layer 2/3 without retraining was examined showing even slightly improved performance for the hand-designed feature representation but a drop of all CNN-based classifiers, which is strongest for the fully learned texture features.

Feature design has a substantial impact on the classification performance of a machine learning algorithm and requires expertise and domain knowledge. Deep learning techniques such as CNNs allow learning features directly from raw data rendering the need for manual feature design unnecessary. The learned representation potentially yields a set of features that is more finely tuned for the particular task than hand-designed features. Furthermore, feature learning allows adapting an algorithm to new datasets by simply supplying additional or novel training data instead of manually changing the feature representation. In particular for connectomic data analysis, feature learning can make the application of the same algorithm to different datasets easier by fine-tuning the underlying features to the novel dataset.

The most relevant texture features of synapses in EM data are the presynaptic vesicle cloud and the PSD, while mitochondria often cause false detections (see subsection 3.4.2). Following the

approach of SyConn (Dorkenwald et al., 2017), multi-class CNNs were trained to detect these structures and exploit their co-occurrence. The evaluation was done on a larger test set to have a greater diversity of ultrastructure and to evaluate the classifier performance at different locations of the dataset (Figure 4.3). The larger test set revealed a substantial drop in performance from a F1 score of 0.884 to 0.858 of the SynEM classifier reported in chapter 3, which partially disappeared when using additional training cubes that contain myelinated processes, parts of somata and a segmentation that had a stronger split rate which was closer to the test set segmentation. This highlights the importance of distributed test sets covering a wide range of ultrastructure at different locations of the dataset. Interestingly, the feature representation based on the output and last hidden layer of a CNN resulted in almost the same classification performance as the representation using only the output layer indicating that the additional information contained in the last hidden layer did not substantially differ from the output layer. While all classifiers using learned features showed improved performance on the dense test set (Figure 4.3), this improvement did not directly transfer to inhibitory synapse classification, where the CNN-based approaches showed a lower classification performance for large parts of the precision-recall curve (Figure 4.6). A main reason for this seemed to be a higher number of FP detections at incidental touches of processes that are split into multiple interfaces due to the employed undersegmentation: If a contact between two neurites is split into multiple interfaces, the CNN-based classifiers often assigned similar scores to each interface, i.e. most of the interfaces overlapping with a synapse had high scores. An incidental touch between processes, however, can result in multiple FP interface detections whereas SynEM often only detects one interface in such cases. Thus, although these errors showed up for the segmentation used in the test set, they would not be visible in a better segmentation potentially resulting in a better performance of the CNN-based classifiers for inhibitory synapses as well. Furthermore, the CNN-based classifiers seemed to be slightly more sensitive to small synapses with little sign of a PSD but vesicles close to the membrane potentially also causing additional FP detections at locations which are often also difficult for expert human annotators. To resolve difficult locations, human annotators often use additional biological prior information such as the existence of other synapses of the same bouton or the identity (axonal, dendritic or glial) of postsynaptic processes to make a decision. A promising direction to increase the performance, in particular for inhibitory synapses, would thus be to add an additional step following interface classification that can include biological information such as comparing all synapses of a bouton or including the process identity based on larger parts of processes and not just the interface location.

The applicability of SynEM to image data obtained by different EM techniques was already shown in subsection 3.4.4. However, even datasets imaged by the same EM technique show variations in imaging quality and image statistics making the application of an algorithm to another dataset non-trivial. In particular, the generalization of interface classifiers to novel datasets without retraining is of interest for practical applications. To determine the performance of interface classifiers on novel SBEM datasets without retraining, interface classifiers that were trained on one SBEM dataset were used to predict synapses on a different SBEM dataset. All examined interface classifiers were able to generalize to a novel dataset using only a simple global raw data

normalization showing only a slight drop in performance for the CNN-based classifiers and even an increase in performance for the fully hand-designed feature representation of SynEM. Interestingly, the drop in performance is largest for the fully learned feature representation whereas it is less for the combination of CNN-based and hand-designed features. The reasons for the inferior performance of CNN-based classifiers compared to SynEM on this data did not seem to have a unique phenotype. A source of FN errors seems to be that the novel dataset contained a large number of small synapses with only few vesicles and only weak signs of PSD which had low scores in the voxelwise CNN prediction thus also causing the interface classifier to miss it. For larger synapses, the CNN classification often worked equally well as on the dataset on which it was trained. This could indicate that the different image statistics or the different resolution caused the CNNs to miss out on small vesicle clouds and PSDs.

The multi-class CNN output for synaptic junctions, mitochondria and vesicle clouds was used as texture features for synapse detection via interface classification, but could be beneficial for other classification tasks as well. Beyond that, the multi-class output itself can be used for biological analyses. For example, Dorkenwald et al. (2017) found that the firing rate of cell types is correlated with the density of vesicle clouds and mitochondria of the corresponding neurites. The output for synaptic junctions corresponds to the area of the PSD of synapses, which is highly correlated to synaptic strength (Harris and Stevens, 1989). In particular for inhibitory synapses or soma synapses, the PSD area could provide a more precise measure of synaptic strength then the total contact area from the interface classification approach. This raises the question as to why the CNN output is not used directly to detect the partners by a simple overlap procedure? Despite the reasons mentioned in Dorkenwald et al. (2017) that a dedicated classification step can combine fractured synapses and resolve overlaps with multiple neurons, the SynEM interface representation proved to be superior to a simple voxel-based overlap (cf. Figure 3.11) and it allows to consider additional features, such as shape descriptors or the hand-designed texture features, to further increase classification performance.

In summary, the usage of learned features in the SynEM interface representation further improved the performance of the interface classification approach. Furthermore, CNNs can be easily fine-tuned to novel datasets or be replaced by different networks rendering an explicit adaption of the feature representation unnecessary.

4.4. Cell Segmentation in EM Data

In this section, CNNs were used to predict membranes between neuronal processes with the aim of segmenting cellular processes, called a volume segmentation, which is the basis for the interface definition of SynEM (subsection 3.3.1).

4.4.1. Network Architectures

All networks used for voxelwise membrane prediction were fully convolutional. A regular CNN architecture with pooling operations such as max-pooling or strided convolutions can be trans-

formed into a translation equivariant FCN by using dilated convolutions in all layers subsequent to pooling layers and setting all strides in the original network to one (see also Zlateski et al., 2016), which is also known as the à trous algorithm in wavelet analysis (Mallat, 1999; Long et al., 2015). More precisely, a dilation ratio $d_l \in \mathbb{N}^k$ is introduced for each layer $l = 1, \ldots, L$, where k is the spatial dimensionality of the input data, i.e. $k = 2$ for images and $k = 3$ for image stacks. Let $s_l \in \mathbb{N}^k$ be the stride of layer l. d_l is calculated recursively with $d_1 = 1$ in all dimensions and the dilation ratio of the next layer is given by the pointwise product of the stride s_l and the current dilation ratio

$$d_{l+1} = d_l \cdot s_l, \tag{4.1}$$

i.e. the dilation ratio is the cumulative product of the network strides. After the dilation ratio was calculated for all layers, all strides in the network are set to one, i.e. $s_l = 1$ for each layer l. All convolutions in the network are replaced by dilated convolution with the corresponding dilation ratios d_l (see Equation 2.7). The dilated convolution corresponds to a sparsification of the convolutional filter by replacing it with a d-regular filter containing $d - 1$ zeros between each non-zero entry. Let $w \in \mathbb{R}^L$ be a one-dimensional filter of length L. The d regular filter $w_d \in \mathbb{R}^{L+(d-1)(L-1)}$ is defined by inserting $d - 1$ zeros between entries of w, i.e.

$$w_d(i) = \begin{cases} w(1+k) & \text{if } i = 1 + kd \text{ for some } k \in \{0, \ldots, L-1\} \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

For $d = 2$ and $w = (w_1, w_2, w_3)$ the d-regular filter with $d = 2$ is given by $w_2 = (w_1, 0, w_2, 0, w_3)$, and analogously for higher dimensional filter.

In addition to default CNN architectures that consist of successive convolutional and pooling layers, a multi-resolution network model was used that processes the input in several parallel pathways at different resolutions. The resolutions were realized by using convolutions with different dilation ratios. This is inspired by the inception layer architecture (Szegedy et al., 2015) and multi-resolution methods such as u-net (Ronneberger et al., 2015; Grais et al., 2017; Zung et al., 2017; Pelt and Sethian, 2018; Chen et al., 2018). Each layer in the multi-resolution network consisted of three parallel pathways operating on the full image resolution (no dilation), a medium resolution (dilation factor 2 in the x and y dimension) and a low resolution (dilation factor of 4 in x and y direction and 2 in z direction) using one or multiple consecutive convolutions with filter sizes $3 \times 3 \times 2$ in the high resolution layer and dilated convolutions with filter sizes $3 \times 3 \times 3$ in the low resolution layers (Figure 4.8). The outputs of pathways with a resolution lower than the image resolution were concatenated to the full resolution pathway as soon as both pathways had the same field of view with respect to the layer input. After the output of the pathway with lowest resolution is merged into the high-resolution pathways, the result is passed through a final convolutional operation that can further reduce or expand the total number of output feature maps and furthermore ensures an anisotropy factor of 2.5 between the first two and the third image dimensions to match the anisotropy of the dataset 2012-09-28_ex145_07x2 (Boergens et al., unpublished; see also subsection 4.4.5). The high-resolution path used residual blocks (He et al., 2016). If the input to the high-resolution pathway had less feature maps than the number of fea-

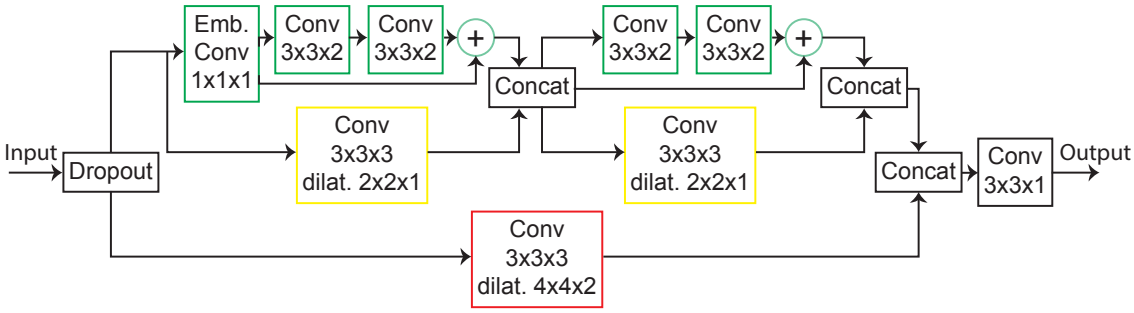

Figure 4.8: Multi-resolution network layer. The input is processed in three parallel pathways characterized by different dilation ratios (dilat.): Full resolution (no dilation, green), medium resolution (dilation factor $2 \times 2 \times 1$, yellow), low resolution (dilation factor $4 \times 4 \times 2$, red) (see also Pelt and Sethian, 2018; Grais et al., 2017). The number of feature maps along a pathway is kept constant. If the number of feature maps of the input is less than the number of feature maps in the high-resolution pathway, the input is passed through an embedding layer (Emb.) to increase its number of feature maps to the number of feature maps in the pathway, otherwise the embedding layer is omitted. A non-linearity is applied after each convolution operation and is not depicted separately. Circles containing a plus indicate the voxelwise sum of two data volumes which is used for residual blocks (He et al., 2016) in the high-resolution pathway.

ture maps along the pathway, then the data was processed by an embedding layer consisting of a $1 \times 1 \times 1$ convolution that increases the number of feature maps. If the number of feature maps in the input to the high-resolution pathway was higher than the number of feature maps along the pathway, only the first feature maps of the input were used in the skip connections and the embedding layer was omitted. Dropout (Srivastava et al., 2014) was applied to the input of each multi-resolution layer and a non-linearity was applied after each convolution operation. After the last multi-resolution layer a $1 \times 1 \times 1$ convolutional layer followed by a non-linearity was used as the output layer. Due do the exclusive usage of convolutional layer, the multi-resolution network described here is fully convolutional and translation equivariant.

Furthermore, a 3D u-net architecture (Ronneberger et al., 2015) similar to Funke et al. (2018) was used (Figure 4.9). The contracting pathway consisted of four different resolutions. For downsampling to a lower resolution, 3D max-pooling layer were used. The first max-pooling layer had a window size of $2 \times 2 \times 1$ to partially account for the data anisotropy and all subsequent max-pooling layer used a window size of $2 \times 2 \times 2$. At each resolution, two successive convolutions were performed with a size of $3 \times 3 \times 3$ at the two highest resolutions and $3 \times 3 \times 3$ and $3 \times 3 \times 1$ at the lower resolutions to account for the data anisotropy. In the expanding pathway, features from lower resolution are upsampled and combined with features from higher resolutions. Upsampling was done using transposed convolutions with filter sizes that equal the corresponding max-pooling window during the contracting pathway. The concatenated features are then further processed by two consecutive 3D convolutions of size $3 \times 3 \times 3$. The output is mapped to the required number of output channels using a $1 \times 1 \times 1$ convolution. Each convolutional operation was followed by a pointwise non-linearity. The resulting u-net architecture had a field of view of $88 \times 88 \times 36$ voxels corresponding to approximately $1\,\mu\mathrm{m}$ in all dimensions (cf. Funke et al., 2018).

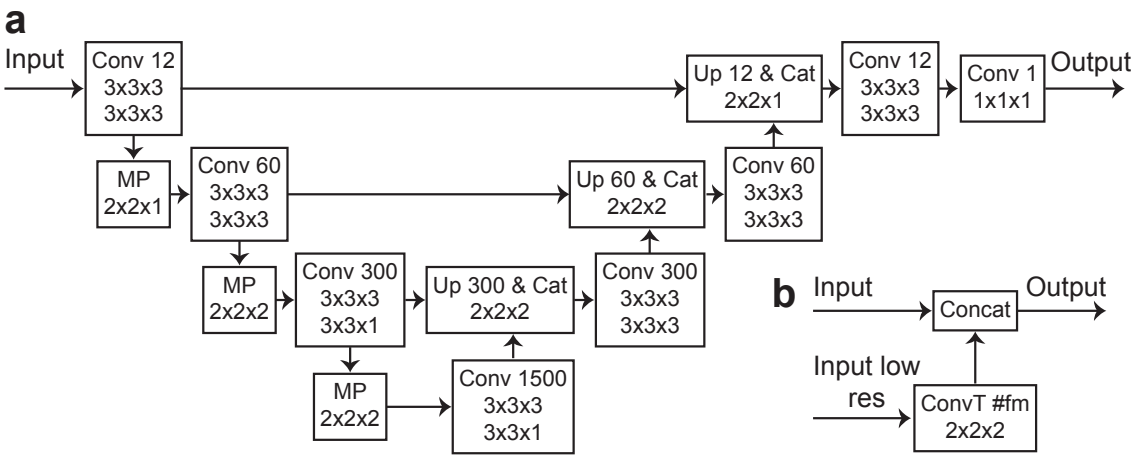

Figure 4.9: U-net architecture. The u-net architecture (Ronneberger et al., 2015) based on Funke et al. (2018) used for membrane prediction. (a) Computation graph for the u-net architecture (arrows correspond to data, boxes to transformations of the data). Convolutional blocks ("Conv #fm") consist of multiple successive convolutions with filter sizes indicated in the corresponding box each followed by a non-linearity with a fixed number of feature maps #fm along the path. Max-pooling layers ("MP") are used to downsample the data to a lower resolution level with non-overlapping max-pooling windows with size indicated in the corresponding box. In the upsampling pathway, features from a resolution level in the donwsampling path are concatenated to upsampled features from a lower resolution ("Up #fm & Cat"). Upsampling is done using transposed convolutions of the specified filter size and #fm output channels followed by a non-linearity. (b) Detailed description of an "Up #fm & cat" transformation depicted in (a) that combines features from a resolution level ("Input") with features from a lower resolution ("Input low res"). The low-resolution data is upsampled using a transposed convolution ("ConvT #fm") and reduced to #fm channels followed by a non-linearity and the result is concatenated with features from the current resolution level.

4.4.2. Network Training

CNN training was done using a custom 3D CNN framework implemented in Matlab and a tensorflow (Abadi et al., 2015) implementation that both used max-filtering and skip-kernels (Sermanet et al., 2014; Long et al., 2015) to efficiently implement CNNs with pooling operations. The Matlab framework implemented convolutions using fast Fourier transform (FFT) adapted for dilated convolutions in the following way. Let $\mathcal{F}_N = (\omega_N^{(k-1)(l-1)})_{k,l=1}^N$ be the matrix representing the discrete Fourier transformation of an N dimensional input, where $\omega_N = \exp(-2\pi i/N)$. If N is divisible by d, then the submatrix $(\mathcal{F}_N)_{k,l}$ for $k = 1, \dots, N/d$ and $l = 1, 1+d, 1+2d, \dots, 1+(N/d-1)d$ is equal to $\mathcal{F}_{N/d}$ by definition of $\mathcal{F}_N$ and $\mathcal{F}_{N/d}$. Furthermore, since

$$\omega_N^{(k+n_1\frac{N}{d}-1)(1+n_2d-1)} = \omega_N^{(k-1)(1+n_2d-1)} \omega_N^{n_1 n_2 \frac{N}{d} d} = \omega_N^{(k-1)(1+n_2d-1)}, \tag{4.3}$$

for $n_1, n_2 \in \mathbb{N}$, the rows of the corresponding columns of $\mathcal{F}_N$ specified above are repeated after d steps. Hence, $(\mathcal{F}_N)_{k,l} = \mathcal{F}_{N/d}$, where $k = \tilde{k}, \dots, \tilde{k} + N/d$ separately for each $\tilde{k} = 1, 1+d, \dots, 1+(N/d-1)d$ and $l = 1, 1+d, 1+2d, \dots, 1+(N/d-1)d$. Thus, the output of $\mathcal{F}_N w_d$ is equal to $\mathcal{F}_{N/d} w$ replicated d times. If the input size was not divisible by d it was enlarged with trailing zeros. All architectures used a scaled $\tanh$ of the form $h(x) = 1.7159 \tanh(0.66x)$ as non-linearity (LeCun et al., 1998) in all layers including the output layer and a sum of squares error (Equation 2.16) normalized to the number of voxels in the output to make the learning rate independent of the batch size. Unless stated otherwise, dropout (Srivastava et al., 2014) was used before each convolutional layer. For the multi-resolution architecture, dropout was only applied to the input of a layer as shown in Figure 4.8. No dropout was used in the u-net architecture.

4.4.3. Volume Segmentation Generation

Volume segmentations were generated from the image complement of the voxelwise membrane predictions using a marker-based watershed algorithm with three different procedures of marker generation. The first two procedures were proposed in SegEM (Berning et al., 2015). In the first procedure, local minima with a depth below θ_{hm} were suppressed and the remaining minima above a size threshold of θ_{ms} voxels were used as markers. In the second procedure, the input was thresholded at θ_{mg} and the resulting connected components above a size threshold of θ_{ms} voxels were used as markers. In the third procedure, the input was thresholded at θ_{dt} and a distance transform was calculated on the binary output. The distance transform was then used as the input to the local minimum procedure above (see also Funke et al., 2018). For the watershed transform itself, the Matlab function `watershed` was used in 3D with the 26 connectivity of voxels (Equation 2.4).

4.4.4. Performance Evaluation Metric

For performance evaluation, split merger rates based on ground truth skeletons were used (Berning et al., 2015). Consider N_t ground truth skeletons, each consisting of a set of nodes x_i, $i = 1, \ldots, N_t$ with a total path length L. Let S be a proposal segmentation consisting of N_s segments with ids $j = 1, \ldots, N_s$. A skeleton-segment overlap matrix A_{ij}, $i = 1, \ldots, N_t$, $j = 1, \ldots, N_s$ was constructed from N_t and S, where each entry contains the number of nodes of skeleton i that lie within the segment j

$$A_{ij} = |\{x_i \cap S^{-1}(j)\}| \in \mathbb{N}_0, \tag{4.4}$$

i.e. each row of A represents a skeleton and each column a segment. To avoid that skeleton nodes are placed in watershed boundaries, the segmentation regions were post-hoc dilated to fill up the whole volume (see also Berning et al., 2015). A binarized version $\hat{A}_{ij}$ of the skeleton-segment overlap matrix was calculated by introducing a node threshold θ_n which was set to one in all experiments, i.e.

$$\hat{A}_{ij} = \begin{cases} 1 & \text{if } A_{ij} \geq \theta_n \\ 0 & \text{otherwise} \end{cases}. \tag{4.5}$$

The number of mergers n_{merger} and splits n_{splits} were calculated from $\hat{A}_{ij}$ as

$$n_{\text{merger}} = \sum_{j=1}^{N_s} \max\left(0, \left(\sum_{i=1}^{N_t} \hat{A}_{ij}\right) - 1\right) \tag{4.6}$$

and

$$n_{\text{splits}} = \sum_{i=1}^{N_t} \max\left(0, \left(\sum_{j=1}^{N_s} \hat{A}_{ij}\right) - 1\right), \tag{4.7}$$

respectively. The average distances between mergers d_m and splits d_s were calculated as $d_m = L/n_{\text{mergers}}$ and $d_s = L/n_{\text{splits}}$ respectively. The inter-error distance (IED) was defined as

$$\text{IED} = \frac{1}{1/d_s + 1/d_m} = \frac{L}{n_{\text{merger}} + n_{\text{splits}}} \tag{4.8}$$

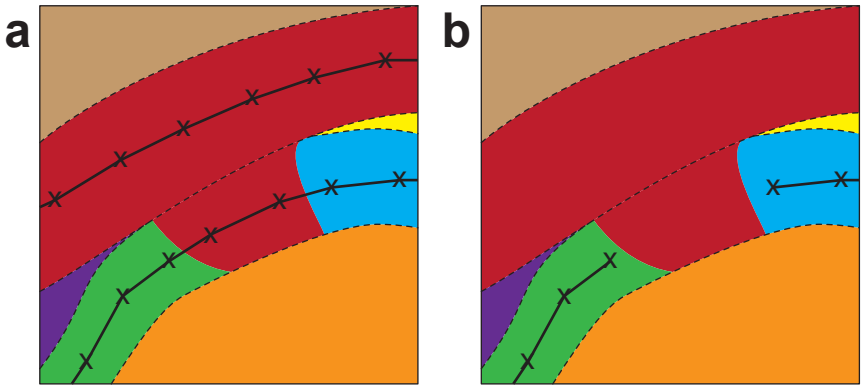

Figure 4.10: Merge-free path length. (a) Illustration of a volume segmentation (different segments displayed in different colors) that contains a merger error (red segment) between two processes for which the cellular membranes are indicated by dashed black lines. For the calculation of the ground truth skeleton path length, SegEM (Berning et al., 2015) used the whole path length of all ground truth skeletons (solid black lines with markers 'x' for skeleton nodes) irrespective of any errors in the proposed segmentation. (b) The merge-free path length is calculated for ground truth skeletons after all nodes in segments that contain nodes from several skeletons (red segment) are deleted. Note that the merge-free path length is only used to calculate the average distance between mergers and thus acknowledges that merge errors have a finite extend in contrast to split errors.

Furthermore, an adapted average path length between mergers $\tilde{d}_m$ was calculated that only considers the ground truth skeleton path length in segments without merge errors (Figure 4.10). Let S_m be the set of segments in a segmentation that contain nodes from at least two skeletons, i.e.

$$S_m = \{i \in \text{im}(S) | \exists k, l : S^{-1}(i) \cap x_k \neq \emptyset \wedge S^{-1}(i) \cap x_l \neq \emptyset\}, \tag{4.9}$$

where $\text{im}(S)$ is the image of S that is all segmentation ids of S. S_m is thus the set of all segments that constitute a merge error. For each ground truth skeleton, all nodes in segments with merger errors were deleted, i.e. the new skeleton nodes were given by

$$\tilde{x}_i = \{x \in x_i | x \notin S^{-1}(S_m)\}. \tag{4.10}$$

Edges to deleted nodes in the skeletons were deleted as well. The total path length of the remaining skeleton nodes is called the merge-free path length $\tilde{L}$. The adapted average distance between mergers was then defined in terms of the merge-free path length $\tilde{L}$ as

$$\tilde{d}_m = \frac{\tilde{L}}{n_{\text{mergers}}}. \tag{4.11}$$

The modified average distance between mergers $\tilde{d}_m$ gives rise to a corresponding adapted IED given by

$$\text{adapted IED} = \tilde{\text{IED}} = \frac{1}{1/d_s + 1/\tilde{d}_m}. \tag{4.12}$$

4.4.5. Experiments

All experiments were done on the dataset 2012-09-28_ex145_07x2 from mouse S1 L4 (Boergens and Helmstaedter, 2012b; see also Berning et al., 2015; Staffler et al., 2017a) using the data from the SegEM segmentation challenge (Berning et al., 2015). The SegEM segmentation challenge consists of volume training data for 243 training regions of size $100 \times 100 \times 100$ voxels each

Name	FOV	Params	Lyr	Filter sizes	Pooling	FMs	Lr	w_{mem}
SegEM	[50, 50, 20]	193641	5	[5x[11, 11, 5]]	-	[4x10, 1]	-	-
cnn2l	[50, 50, 20]	287593	9	[[7, 7, 3], [5, 5, 3], [4, 4, 2], 2x[5, 5, 3], 2x[5, 5, 2], 2x[3, 3, 2]]	4: [2, 2, 2]	[4x24, 4x32, 1]	1e−4	6, 12, 6
cnn17l	[78, 78, 28]	330401	8	[[8, 8, 4], 2x[5, 5, 3], 3x[3, 3, 3], 2x[3, 3, 2]]	2: [2, 2, 1], 4: [2, 2, 2], 6: [2, 2, 2]	[2x16, 2x32, 3x64], 1]	1e−4	12
CD	[56, 56, 22]	329473	14	[8x[5, 5, 3], 6x[5, 5, 2]]	-	[9x16, 4x32, 1]	1e−5	12
CD2	[66, 66, 26]	525913	16	[[7, 7, 3], 9x[5, 5, 3], 6x[5, 5, 2]]	-	[5x16, 5x24, 5x32, 1]	1e−5	6, 10
CD3	[74, 74, 30]	516273	20	[[7, 7, 3], 9x[5, 5, 3], 6x[5, 5, 2], 4x[3, 3, 2]]	-	[7x16, 6x24, 6x32, 1]	1e−5	6, 12
mr_1d	[60, 60 24]	333457	7	6 mr layer & output layer	-	[6x16, 1]	1e−4	6, 12
mr_1e	[60, 60 24]	366893	7	6 mr layer & output layer	-	[6x20, 1]	1e−2	6, 12
mr_1f	[60, 60 24]	467201	7	6 mr layer & output layer	-	[6x32, 1]	1e−3	6, 12
mr_1g	[60, 60 24]	873121	7	6 mr layer & output layer	-	[6x[32, 16, 16], 1]	1e−3	12
mr_2a	[70, 70 28]	391185	8	7 mr layer & output layer	-	[7x16, 1]	1e−2	6, 12

mr_2d	[70, 70 28]	430961	8	7 mr layer & output layer	-	[7x20, 1]	1e−2	6, 12
mr_2e	[70, 70 28]	470737	8	7 mr layer & output layer	-	[7x24, 1]	1e−2	6, 12
mr_3c	[50, 50 20]	384113	6	5 mr layer & output layer	-	[5x32, 1]	1e−4	6, 12
unet_1a	[88, 88, 36]	4116961	-	see Figure 4.9	-	[32, 64, 128, 256]	1e−3	12
unet_1b	[88, 88, 36]	4116961	-	see Figure 4.9	-	[32, 64, 128, 256]	1e−3	12
unet_2a	[88, 88, 36]	1029873	-	see Figure 4.9	-	[16, 32, 64, 128]	1e−2	12
unet_2b	[88, 88, 36]	1029873	-	see Figure 4.9	-	[16, 32, 64, 128]	1e−3	12
unet_3a	[88, 88, 36]	560673	-	see Figure 4.9	-	[16, 32, 48, 64]	1e−2	12
unet_3b	[88, 88, 36]	560673	-	see Figure 4.9	-	[16, 32, 48, 64]	1e−3	12
unet_4a	[88, 88, 36]	45161869	-	see Figure 4.9	-	[16, 60, 300, 1500]	1e−3	12
unet_4b	[88, 88, 36]	45161869	-	see Figure 4.9	-	[16, 60, 300, 1500]	1e−2	12
unet_4c	[88, 88, 36]	45161869	-	see Figure 4.9	-	[16, 60, 300, 1500]	1e−3	12

Table 4.3: **CNN architectures used for membrane prediction.** Name: Internal name of the network architecture; FOV: 3D field of view in voxels; Params: Total number of trainable parameters; Lyr: Number of network layers where a multi-resolution (mr) layer and the combination of a convolution and a pooling operation are considered as a single layer; Filter sizes: Spatial size of the filters in each layer in voxels, consecutive layers with the same filter shape are denoted by $n\times$ filter size; Pooling: Layer number and spatial size of the max-pooling window; FMs: Feature maps for each layer, consecutive layers with the same number of feature maps are denoted by $n\times$ feature map. For multi-resolution networks, the feature maps are either specified separately for each of the three pathways (mr_net_1g) or as a single number for all pathways (all other mr_net architectures). Lr: Learning rate used during training; w_{mem}: Weight for the membrane class, multiple numbers correspond to consecutive training runs.

with labels for membrane and intracellular voxels and the surrounding raw EM data with a total border of $100 \times 100 \times 50$ voxels. Furthermore, it contains two regions where each cellular process is traced by a skeleton, which were used for hyperparameter search and performance evaluation. Since the membrane segmentation of the volume training data was artificially enlarged for SegEM causing small processes to vanish almost completely, the original volume tracings of processes that were generated by contouring were used, which mark cellular processes with positive ids (received from the SegEM authors upon personal request). All voxels in the volume tracings that did not belong to any process, i.e. had an id of zero, were considered as membrane voxels. To introduce a boundary between directly adjacent processes, all voxels that belong to a process but that also have the id of a different process in their 8 neighborhood in the x-y plane (2D) were considered membrane voxels as well. Target values for membranes were set to -1 and for non-membranes to 1. A weight of 6 to 12 was used for the membrane class to bias the subsequent watershed segmentation step towards splits. The input data was augmented using random rotations by multiples of 90 degree in the x-y plane and random mirroring along the z dimension. Network training was done until no significant change in the training loss was observed anymore which typically took 8 to 16 days on a single GPU. To compare to the results of Funke et al. (2018), the architectures unet_4a, unet_4b, unet_4c were used, which closely resemble their architecture. Unet_4a was trained using the same optimizer and hyperparameter settings as reported by Funke et al. (2018) whereas unet_4b and unet_4c were trained using stochastic gradient descent with momentum. The trained network architectures are summarized in Table 4.3. Ensemble of the trained networks were considered by combining the membrane predictions of different networks using a voxelwise mean or minimum operation (Table 4.4).

The evaluation of volume segmentations was done using split and merger rates on the ground truth skeletons provided by SegEM. For each classifier, the membrane predictions were calculated for the regions of the training and test skeletons. Segmentations were generated from the boundary maps using three different segmentation procedures based on the watershed algorithm as described in subsection 4.4.3. The skeleton training set was used for hyperparameter search of the watershed procedure and the evaluation of split and merger rates was done on the skeleton test set. The watershed parameters that resulted in the largest adapted IED (Equation 4.12) were individually determined for each network architecture using a grid search. The adapted IED was used for this due to the treatment of merger errors in the IED proposed in SegEM that can decrease the error when merging two segments (see also Januszewski et al., 2017). Afterwards, the segmentation procedure (consisting of a network architecture and the previously determined watershed parameters) with the highest IED on the skeleton training set was determined. For SegEM, the segmentation parameters for the optimal IED determined in Berning et al. (2015) were used. The optimal split-merge results on the skeleton training set are summarized for the individual networks in Table 4.5 and for the ensembles in Table 4.6. For all networks, the best segmentation was obtained using the first watershed procedure based on local minima suppression and small object removal. The best network (mr_net_1g) and the best ensemble (ens_mr_3) were evaluated on the skeleton test set and compared to SegEM (Table 4.7). Note that the best ensemble even resulted in a higher IED than the approach of Funke et al. (2018) (see Table 1 in

Name	Mode	No	Networks
ens_CD	mean	3	CD, CD2, CD3
ens_CD_min	min	3	CD, CD2, CD3
ens_mr_1	mean	3	mr_1g, mr_2e, mr_3c
ens_mr_1_min	min	3	mr_1g, mr_2e, mr_3c
ens_mr_2	mean	3	mr_1g, mr_1e, mr_1f
ens_mr_2_min	min	3	mr_1g, mr_1e, mr_1f
ens_mr_3	mean	8	mr_1d, mr_1e, mr_1f, mr_1g, mr_2a, mr_2d, mr_2e, mr_3c
ens_mr_3_min	min	8	mr_1d, mr_1e, mr_1f, mr_1g, mr_2a, mr_2d, mr_2e, mr_3c
ens_1	mean	6	CD, CD2, CD3, mr_1g, mr_2d, mr_3c
ens_1_min	min	6	CD, CD2, CD3, mr_1g, mr_2d, mr_3c
ens_2	mean	5	unet_1a, unet_1b, mr_1g, mr_2d, mr_3c
ens_2_min	min	5	unet_1a, unet_1b, mr_1g, mr_2d, mr_3c

Table 4.4: **Ensembles of CNN architectures used for membrane prediction.** Name: Internal name of the ensemble; Mode: Mode of combination of the single CNN outputs which was either a mean or minimum operation for each voxel. Note that the target label is -1 for membrane voxels and 1 otherwise motivating the use of the minimum operation. No: Number of networks in the ensemble; Networks: Names of the individual networks in the ensemble according to Table 4.3.

Funke et al., 2018) although no agglomeration procedure was used. The split-merger curves on the test set are shown in Figure 4.11. The membrane prediction output and the segmentation for a single test set section is displayed in Figure 4.12.

4.4.6. Discussion

Fully convolutional networks (FCNs) were trained to predict membranes between cellular processes as part of the volume segmentation procedure for EM data suggested by SegEM (Berning et al., 2015). A novel multi-resolution architecture was proposed that uses dilated convolutions instead of subsampling to process the input without reducing the spatial density of the predictions. The multi-resolution architecture was compared to regular CNN architectures, which are built from successive convolutional and pooling layers, and to the u-net architecture (Ronneberger et al., 2015), which uses an encoder-decoder structure. Despite having substantially less parameters, the multi-resolution networks were found to perform better than different variants of a u-net architecture similar to Funke et al. (2018), which was proposed for volume segmentation in connectomics using affinity maps. Almost all networks were found to outperform SegEM by a substantial margin of a factor larger than 2 in the IED metric on the SegEM segmentation challenge. The overall best result was achieved using an ensemble of multiple networks.

Name	θ_{hm}, θ_{ms}	d_m	$\tilde{d}_m$	d_s	ad. IED	IED
mr_net_1g	0.27, 10	18.51	10.15	5.67	3.64	4.34
mr_net_2e	0.51, 0	17.72	9.15	5.37	3.38	4.12
mr_net_2d	0.41, 0	15.59	7.77	5.49	3.22	4.06
mr_net_1e	0.27, 100	18.85	10.67	5.13	3.47	4.03
mr_net_1f	0.33, 0	19.38	10.78	5.05	3.44	4.00
CD3_2	0.27, 100	16.46	9.10	5.21	3.31	3.96
unet_1b	0.43, 0	21.60	13.36	4.71	3.48	3.87
mr_net_3c	0.29, 100	19.75	11.71	4.79	3.40	3.85
unet_1a	0.35, 0	18.03	9.10	4.83	3.16	3.81
mr_net_2a	0.27, 100	16.72	8.95	4.83	3.14	3.75
unet_3b	0.31, 10	17.43	9.28	4.59	3.07	3.63
CD2_2	0.39, 0	16.46	8.55	4.65	3.01	3.63
unet_2b	0.27, 100	18.03	10.35	4.51	3.14	3.61
unet_3a	0.39, 0	16.86	8.45	4.55	2.96	3.58
mr_net_1d	0.35, 0	16.86	8.79	4.46	2.96	3.53
unet_4c	0.23, 50	17.43	9.08	4.41	2.97	3.52
unet_2a	0.23, 0	15.83	7.67	4.07	2.66	3.24
CD_1	0.35, 100	16.33	9.26	3.96	2.77	3.19
unet_4b	0.29, 50	15.25	7.86	3.91	2.61	3.11
cnn2l_14	0.59, 0	16.46	10.01	2.76	2.16	2.36
unet_4a	0.10, 50	8.68	3.11	2.39	1.35	1.87
cnn17l_3	0.41, 0	12.27	6.82	1.86	1.46	1.61
segEM	0.58, 50	9.38	5.03	1.49	1.15	1.29

Table 4.5: **Volume segmentation validation set results.** Results are sorted by decreasing IED. Name: Name of the CNN according to Table 4.3; θ: Parameters for local minima based watershed procedure; d_m: Average distance between mergers in µm; $\tilde{d}_m$: Adapted average distance between mergers in µm; d_s: Average distance between splits in µm; ad. IED (µm): Adapted inter-error distance in µm. IED: Inter-error distance in µm;

Name	θ_{hm}, θ_{ms}	d_m	$\tilde{d}_m$	d_s	ad. IED	IED
ens_mr_3	0.57, 0	18.68	10.64	7.08	4.25	5.13
ens_mr_1	0.53, 50	20.13	11.56	6.60	4.20	4.97
ens_2	0.51, 0	22.54	13.43	6.15	4.22	4.83
ens_1	0.49, 0	19.94	11.49	6.25	4.05	4.76
ens_mr_1_min	0.65, 0	22.30	13.65	5.91	4.12	4.67
ens_mr_3_min	0.80, 0	22.79	13.97	5.87	4.14	4.67
ens_mr_2	0.37, 0	21.83	13.28	5.79	4.03	4.58
ens_1_min	0.69, 0	24.11	15.64	5.62	4.13	4.56
ens_mr_2_min	0.45, 0	25.92	16.83	5.47	4.13	4.52
ens_2_min	0.67, 0	24.69	15.79	5.41	4.03	4.44
ens_CD_min	0.59, 0	15.36	7.67	5.70	3.27	4.16
ens_CD	0.37, 0	19.38	10.95	4.90	3.39	3.91

Table 4.6: **Volume segmentation ensemble validation set results.** Results are sorted by decreasing IED. Name: Name of the ensemble according to Table 4.4; θ: Parameters for local minima based watershed procedure; d_m: Average distance between mergers in µm; $\tilde{d}_m$: Adapted average distance between mergers in µm; d_s: Average distance between splits in µm; ad. IED (µm): Adapted inter-error distance in µm. IED: Inter-error distance in µm;

Name	d_m	$\tilde{d}_m$	d_s	ad. IED	IED
ens_mr_3	15.58	10.96	7.10	4.31	4.88
Funke et al. (2018)	21.34	-	6.27	-	4.84
mr_net_1g	13.06	8.27	5.96	3.47	4.09
SegEM (Berning et al., 2015)	3.48	-	2.89	-	1.58
SegEM (redo)	3.50	1.02	1.90	0.66	1.23

Table 4.7: **Volume segmentation test set results.** Results are sorted by decreasing IED. Name: Name of the network/ensemble according to Table 4.3 and Table 4.4 or reference of the corresponding publication (Berning et al., 2015; Funke et al., 2018). SynEM (redo) refers to the results that was achieved in a reproduction of the SegEM results using the same segmentation parameters as specified in Berning et al. (2015) without the moving average calculation. θ: Parameters for local minima based watershed procedure; d_m: Average distance between mergers in µm; $\tilde{d}_m$: Adapted average distance between mergers inµm; d_s: Average distance between splits in µm; ad. IED (µm): Adapted inter-error distance in µm. IED: Inter-error distance in µm;

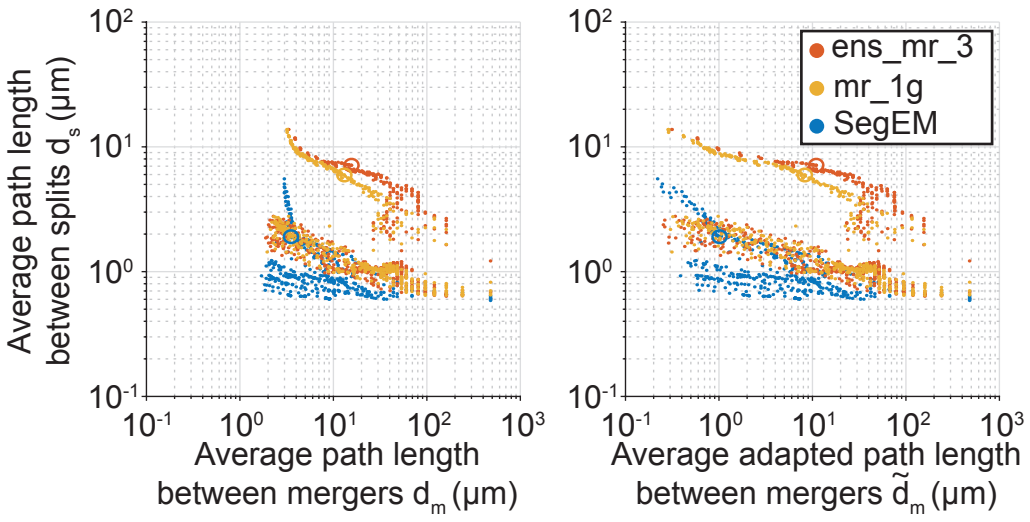

Figure 4.11: Volume segmentation test set result. Split-merge curves of the best ensemble of networks (mr_mr_3; red) and the best single network (mr_1g; yellow) in comparison to SegEM (blue; Berning et al., 2015) with respect to the average path length between mergers (left) and the adapted average path length between mergers (right). The segmentations with optimal IED according to the training set are marked by circles (ens_mr_3: 4.88 µm; mr_1g: 4.09 µm; SegEM: 1.23 µm).

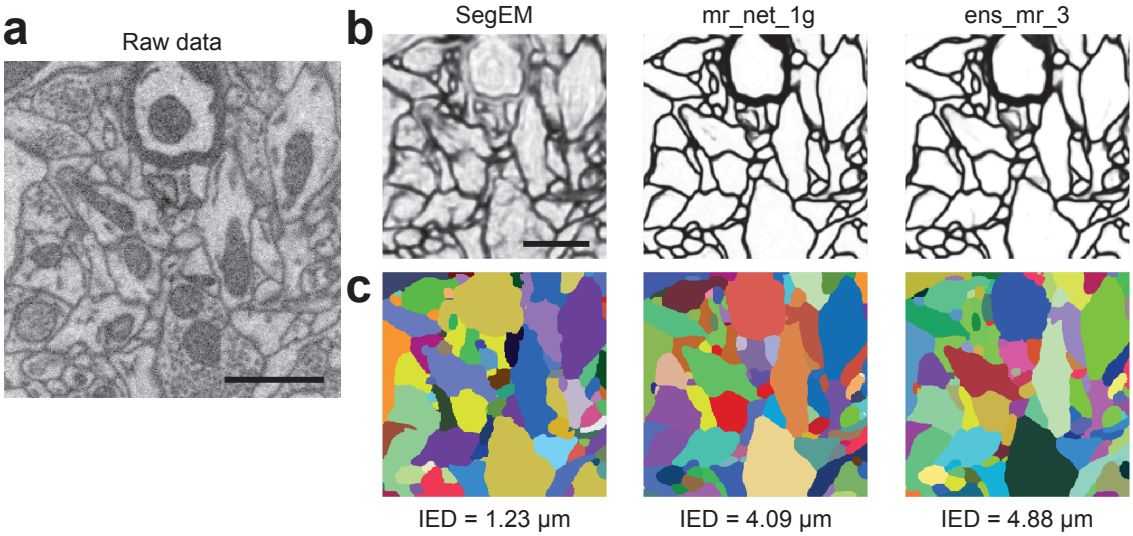

Figure 4.12: Volume segmentation examples. (a) Section of the raw data from the SegEM skeleton test set ($z = 150$; Berning et al., 2015). (b) Membrane predictions for SegEM (left), mr_net_1g (middle) and ens_mr_3 (right). (c) Segmentation at the optimal segmentation parameters found on the skeleton training set for the corresponding prediction in (b) (see also Table 4.5 and Table 4.6). Scale bars: 1 µm (a, b). Scale bar in (b) applies to all images in (b) and (c).

Techniques from deep learning, in particular in the form of FCNs, have provided remarkable advances in semantic segmentation (Long et al., 2015; Ronneberger et al., 2015; Chen et al., 2018). However, commonly used encoder-decoder network architectures do not produce translation equivariant outputs and typically use a high number of trainable parameters (Ronneberger et al., 2015). The multi-resolution architecture presented here is not based on an encoder-decoder structure but uses multiple resolutions in each layer of the network realized through dilated convolutions resulting in a dense output that is translation equivariant. The architecture combines ideas from Pelt and Sethian (2018) and Grais et al. (2017) and is easy to implement in existing deep learning frameworks such as tensorflow (Abadi et al., 2015). The shortest flow of information from input to output through the network along low-resolution pathways provides spatial context for semantic classification which is frequently merged with the high-resolution pathway allowing the network to provide the necessary localization and to choose the most appropriate resolution for further processing. The proposed approach merges pathways only if they have the same field of view thus limiting the overall field of view of the network by the high-resolution pathway. An alternative would be to consider less successive convolutions in the high-resolution pathways and crop their outputs to the size of a lower-resolution pathway before merging them. In the most extreme case, only one convolution for each dilation rate could be used followed by a concatenation similar to Grais et al. (2017). The number of parameters in a multi-resolution network that resulted in a similar performance as commonly used encoder-decoder architectures (Ronneberger et al., 2015; Funke et al., 2018) was substantially lower as also noted by Pelt and Sethian (2018). The computational cost on the other hand is typically higher since feature maps are not explicitly downsampled. For SBEM data, however, the original data resolution already compromises high spatial information with a larger field of view, which makes it easier to achieve sufficient classification context without exceedingly deep networks, motivating network architectures with a high focus on the native data resolution that are able to detect the smallest processes.

The volume segmentation underlying the SynEM interface definition (subsection 3.3.1) was provided by SegEM and previous experiments indicated that a better segmentation improves the performance of SynEM (subsection 3.4.2). Beyond potential benefits for synapse detection, the generation of high quality volume segmentations is crucial for the automation of the connectomic reconstruction (see section 1.2). The SegEM segmentation procedure for the cortex dataset of the SegEM challenge consists of a membrane prediction step using a CNN followed by a watershed segmentation to produce an initial oversegmentation of the EM data, which is similar to other segmentation procedure used in connectomics (Funke et al., 2018). The CNN used in SegEM consisted of five convolutional layers with large filter sizes (11, 11, 5 in the x, y, z dimension, respectively). Since much progress has been made in semantic segmentation using more refined network architectures, the performance of the SegEM segmentation procedure for different network architectures was evaluated. No additional agglomeration was done subsequent to the watershed step which is expected to further improve the segmentation quality substantially (Funke et al., 2018). Segmentation quality was measured using the IED proposed by Berning et al. (2015), which compares a proposal segmentation with a dense skeletonization of all processes in a test region. However, similar to Januszewski et al. (2017), the IED was found to

be problematic because merging two segments can decrease the total number of merger errors and thus increase the average distance between mergers. To counteract this behavior of the IED, which is strongest in an undersegmentation regime, the merge-free path length was introduced. For the calculation of the merge-free path length, the path length in segments with merge errors is subtracted from the total path length. The merge-free path length gives rise to an adapted average distance between mergers and an adapted IED. The adapted average distance between mergers is bounded from below by zero and provides a more intuitive inter-error distance by acknowledging the finite extend of merger errors in contrast to split errors. While the adapted IED does not modify the counting procedure of merger errors in the IED, it typically resulted in split-biased segmentations in practice. Thus, the adapted IED was used to select the best watershed parameters on the skeleton training set separately for each network architecture (Table 4.5). After the best watershed parameters were determined for each network, the IED was used to select between the networks on the skeleton training set for performance comparison on the skeleton test set (Table 4.7). Architectures with less than 10 layers that employ pooling to increase their field of view showed only minor improvements over the SegEM baseline (cnn2l and cnn17l). Deeper networks with 14 or more layers and without any pooling operations were already able to achieve good results (CD_1, CD2_2, CD3_2) which were even better than the u-nets on the skeleton training set. Interestingly, the architecture unet_4a, which was designed close to Funke et al. (2018) and trained exactly as they described (using the Adam optimizer; Kingma and Ba, 2014), showed a remarkably similar performance to their baseline model whereas unet_4c, which had the same architecture but a different optimizer (stochastic gradient descent with momentum), resulted in a much better performance. The multi-resolution architecture proposed here was able to outperform the other architectures providing the best result of a single network with an IED of $4.09\,\mu\mathrm{m}$, which is more than a factor 2.5 higher than the IED of $1.58\,\mu\mathrm{m}$ of SegEM (see Table 2 in Berning et al., 2015). To achieve even better segmentation results, ensembles of networks were used as they have been shown to typically outperform single networks (Hansen and Salamon, 1990). Ensembles of networks can be constructed for example by averaging the output of different networks (Ciresan et al., 2012; Drozdzal et al., 2018) or using the same network on slightly altered inputs such as rotated versions of the same field of view (Ronneberger et al., 2015). Here, the outputs of different architectures were combined by voxelwise averaging or a voxelwise minimum operation. The performance of each ensemble was found to be better than the performance of the individual networks. The overall best result was achieved by an ensemble of eight networks resulting in an IED of $4.88\,\mu\mathrm{m}$ which is more than a factor of 3 higher than for SegEM and comparable to the result of Funke et al. (2018) even without any agglomeration subsequent to the watershed step.

The segmentation approach used here is fundamentally impeded by the sequential two-step process of boundary prediction and watershed partitioning. Even a single misclassified voxel along a membrane can result in a merge error which is often dealt with by using a distance transform watershed in the partitioning step (Beier et al., 2017; Funke et al., 2018). Conversely, the training procedure used here consisting of a voxelwise sum of squares error does not directly optimize the membrane predictions for the subsequent watershed step which requires specific loss functions (see for example Turaga et al., 2009; Jain et al., 2010a; Kroeger et al., 2013). Split errors occur

if two seeds are placed within the same process requiring a subsequent agglomeration step. Furthermore, the employed watershed procedure of the SegEM approach results in cellular processes that are separated by a one-voxel thick boundary which can be problematic for thick membranes and frequently causes merge errors close to myelin (see Figure 4.12). Relaxing this condition by stopping the watershed at a maximal value could prevent errors in such cases but would require an adaption of subsequent processing steps such as the SynEM interface definition that requires a one-voxel thick boundary. Flood-filling networks (Januszewski et al., 2017) are a recently proposed approach that combines membrane prediction and partitioning into a single algorithm that can also naturally be used for agglomeration. A flood-filling network consist of a recursive CNN that iteratively refines a binary segmentation mask corresponding to a single process. The classifier is trained end-to-end and intrinsically results in a large penalty for merge errors since all voxels in the reconstructed objects contribute to the error.

In summary, more refined network architectures are able to improve upon the SegEM segmentation result by a substantial margin. A novel multi-resolution architecture was able to outperform a previously proposed u-net architecture (Ronneberger et al., 2015; Funke et al., 2018) and regular CNNs on medium-resolution SBEM data.

4.5. Conclusion

In conclusion, it was shown that fully convolutional networks (FCNs) can further improve the performance of SynEM as well as the segmentation provided by SegEM (Berning et al., 2015) underlying the interface definition of SynEM. Replacing the hand-designed features of SynEM with features learned by a FCN does not only improve performance over the hand-designed features but also renders the need for feature design unnecessary. In addition, despite the inferior performance on a new dataset without retraining, FCNs offer a straightforward strategy to fine-tune the representation to new datasets. The proposed multi-resolution architecture was shown to outperform other architectures for membrane detection on a medium resolution SBEM dataset. The substantial improvements for volume segmentations over SegEM, although likely beneficial for synapse detection as well, are particularly relevant for the dense reconstruction of large datasets. Only if the average distance between errors in the volume segmentation is large enough such that focused annotation is faster than dense skeletonization, the progress in segmentation methods can actually be leveraged to reduce manual annotation time.

5. Application to Circuit Reconstruction

In this chapter, the SynEM interface classification method was used to analyze all synapses innervating the dendritic tree of a spiny stellate neuron in layer 4 (L4) of mouse primary somatosensory cortex (S1). The distribution of sizes of synaptic contacts is shown to be well described a lognormal distribution for all synapses as well as for spine and shaft synapses separately. The distribution of distances from synapses along the dendrites to the soma indicates that shaft synapses are formed more proximally than spine synapses.

5.1. Introduction and Related Work

Synaptic plasticity, which refers to the change of strength of synapses over time due to variations in their activity, has long been related to learning and memory (Hebb, 1949; Lynch, 2004; Whitlock et al., 2006). The distribution of synaptic weights should thus contain information about the underlying plasticity and learning rules (Barbour et al., 2007). Electrophysiological recordings have shown that the distribution of synaptic strength, defined as the peak excitatory postsynaptic potential amplitude, follows a lognormal distribution (Song et al., 2005, see also Buzsaki and Mizuseki, 2014). The synaptic strength is highly correlated with structural properties of synapses such as the area of the postsynaptic density (PSD) (Harris and Stevens, 1989), the area of the active zone (Schikorski and Stevens, 1997) and the axon-spine interface area (ASI) for excitatory synapses (Cheetham et al., 2012) which can be investigated in electron microscopy (EM) datasets. Several studies have investigated the distributions of synapse sizes for all synapses in EM datasets showing that they also follow a lognormal distribution and that shaft synapses are significantly larger than spine synapses (Merchán-Pérez et al., 2014; Santuy et al., 2018).

Here, the distribution of synapse sizes was examined for all synapses onto the dendritic tree of a spiny stellate neuron in a serial block-face electron microscopy (SBEM) dataset from mouse S1 L4. Unlike previous studies, the synapses share a common postsynaptic partner and thus correlate with the distribution of input weights onto a spiny stellate neuron. It is shown that also in the case of synapses with a common postsynaptic partner, the distribution of synapse sizes follows a lognormal distribution for all synapses as well as for shaft and spine synapses separately. Furthermore, the distribution of path length from synapses to the soma shows that shaft synapses are placed more proximal to the soma along the dendritic tree compared to spine synapses, which is in accordance to previous studies (Anderson et al., 1994; Kornfeld et al., 2017).

5.2. Methods

The SBEM dataset 2012-09-28_ex145_07x2 from mouse S1 L4 (Boergens and Helmstaedter, 2012b; see also Berning et al., 2015; Staffler et al., 2017a) and the SegEM segmentation (Berning et al., 2015) in a centered bounding box of size $86 \times 52 \times 86\,\mu\mathrm{m}^3$ calculated for the SynEM

test set was used (see subsection 3.4.1). The border area of each interface was calculated as suggested in Berning et al. (2015). SynEM was applied to the whole dataset to generate synapse scores for each interface. A spiny stellate neuron with a soma location roughly in the center of the dataset (soma bounding box [4700, 6300; 2500, 3600; 1200, 1800]) was selected and reconstructed by skeletonization. The skeleton was split into separate trees for the dendritic and axonal processes. The tree corresponding to the axonal process was discarded and all segments of the SegEM segmentation were collected which contained at least one node of the skeleton. Afterwards, all interfaces between the segments of the spiny stellate neuron and any other segment were extracted. A list of segments in the soma of the spiny stellate neuron was available and interfaces between skeleton segments and segments contained in the soma were discarded. For synapse detection, the focused proofreading strategy proposed in subsection 3.4.5 was applied using the SynEM test set of 2012-09-28_ex145_07x2 to determine the thresholds for synapse detection. To determine the thresholds for focused proofreading, all non-synaptic interfaces of the test set were used. For each synapse of the test set, only the interface with highest score overlapping with the synapse was kept. Then, the highest possible score θ_1 with a single synapse recall of 95% and the lowest possible score θ_2 with a synapse precision of 95% were determined. The interfaces of the spiny stellate with a score above θ_2 were labeled as synaptic. 2025 interfaces had a score in the range $[\theta_1, \theta_2]$ and were proofread by an expert neuroscientist using webKnossos (Boergens et al., 2017) taking a total annotation time of 387 minutes (6 hours and 27 minutes) corresponding to 11.5 seconds per interface. In addition, all interfaces with an area larger than $0.35\,\mu\mathrm{m}^2$ were also manually proofread in webKnossos which applied to 165 interfaces taking a total annotation time of 10 minutes (3.6 s per interfaces). After focused annotation, a total of 2327 interfaces of the spiny stellate neuron were labeled as synaptic.

The skeleton of the spiny stellate neuron was used to classify synaptic interfaces as spine or shaft synapses. For shaft synapse extraction, all spines were deleted from the spiny stellate skeleton by eroding all degree one nodes of the skeleton until a degree three node was reached. Degree one nodes inside the soma bounding box were not considered for spine identification. All segments that contained a skeleton node after erosion of the degree one nodes were considered as shaft segments. The segments of the spiny stellate neuron that were not labeled as shaft segments were labeled as spine segments. Synaptic interfaces onto a spine segment of the spiny stellate neuron were labeled as spine interfaces while all others as were labeled as shaft interfaces.

To account for synapses split into multiple interfaces, the 2327 synaptic interfaces were clustered using hierarchical clustering with single linkage and a distance cutoff of $350\,\mathrm{nm}$ resulting in a total number of 1334 interface clusters which were called synapses. Three of the resulting synapses contained both shaft and spine interfaces and were manually assigned to the correct class resulting in 970 spine and 364 shaft synapses. A second clustering was done separately for spine and shaft interfaces with a distance threshold of $450\,\mathrm{nm}$ for spine interfaces and $500\,\mathrm{nm}$ for shaft interfaces resulting in 880 spine synapses and 285 shaft synapses. The border area for synapses, also called synapse size, was calculated by summing up the area of all interfaces in the cluster. The comparison of the area distributions of spine and shaft synapses was done using a

two-sided Kolmogorov-Smirnov test using the Matlab `kstest2` function. The distributions of synapse sizes for all synapses and for spine synapses and shaft synapses separately were fitted individually with a lognormal distribution using the Matlab `fitdist` function. One-sample Kolmogorov-Smirnov tests were performed to compare the empirical area distributions and the fitted lognormal distributions.

To calculate the distance from a synapse to the soma, all skeleton nodes in any presynaptic segment of all interfaces in a synapse cluster were considered. For each postsynaptic skeleton node of a synapse, the shortest path to any skeleton node in the soma bounding box was calculated. The distance from a synapse to the soma was then calculated as the mean over all the shortest paths. Synapses for which all postsynaptic skeleton nodes were inside the soma bounding box were discarded. For statistical comparison of the distributions of distances from synapses to the soma for shaft and spine synapses, a two-sided t-test and the Wilcoxon rank sum test were done both using the Matlab functions `ttest2` and `ranksum`, respectively.

5.3. Results

SynEM was used to analyze all synapses onto the soma and dendritic tree of a spiny stellate neuron in the SBEM dataset 2012-09-28_ex145_07x2 from mouse S1 L4 (Boergens and Helmstaedter, 2012b) segmented by SegEM (Berning et al., 2015). The spiny stellate neuron was reconstructed using webKnossos (Boergens et al., 2017) and all segments of the SegEM segmentation that overlapped with a skeleton node were collected. All interfaces between the segments of the spiny stellate neuron and any other process were classified by SynEM and grouped into spine and shaft segments using the skeleton tracing. The focused proofreading strategy of SynEM proposed in subsection 3.4.5 (see also Figure 3.18) was used with the thresholds θ_1 at 95% recall and θ_2 at 95% precision according to the SynEM test. Synaptic interfaces with a score in the range $[\theta_1, \theta_2]$ were proofread while synapses with a score above θ_2 were automatically accepted as synaptic resulting in expected synapse detection precision and recall rates of 95%. Synaptic interfaces that were closer than $350\,\mathrm{nm}$ were clustered to merge synaptic interfaces overlapping with the same synapse. The resulting synapse input map of the spiny stellate contained 1334 synapses out of which 970 (72.7%) were onto spines and 364 (27.3%) onto shafts (Figure 5.1). The interface area was calculated for each synaptic interface and aggregated for all interfaces of a synapse cluster to yield the synapse sizes (Figure 5.2). An average synapse size of $0.27 \pm 0.20\,\mu\mathrm{m}^2$ (mean $\pm$ s.d.; range $0.02\,\mu\mathrm{m}^2$–$1.55\,\mu\mathrm{m}^2$, $n = 1334$) was found (Figure 5.2a). The average size of spine synapses was $0.24 \pm 0.17\,\mu\mathrm{m}^2$ (mean $\pm$ s.d.; range $0.02\,\mu\mathrm{m}^2$–$1.32\,\mu\mathrm{m}^2$, $n = 970$) and of shaft synapses $0.34 \pm 0.26\,\mu\mathrm{m}^2$ (mean $\pm$ s.d.; range $0.02\,\mu\mathrm{m}^2$–$1.55\,\mu\mathrm{m}^2$, $n = 364$) (Figure 5.2a). The distributions of spine and shaft synapse sizes are significantly different ($p < 10^{-9}$, two-sample Kolmogorov-Smirnov test). The distributions of synapse sizes could be well fitted with a lognormal distribution for all synapses ($p = 0.25$, one-sample Kolmogorov-Smirnov test, see Figure 5.2b) as well as for spine synapses ($p = 0.46$, one-sample Kolmogorov-Smirnov test, see Figure 5.2c) and shaft synapses ($p = 0.35$, one-sample Kolmogorov-Smirnov test, see Figure 5.2d) separately. This result was stable for different clustering distance thresholds for both

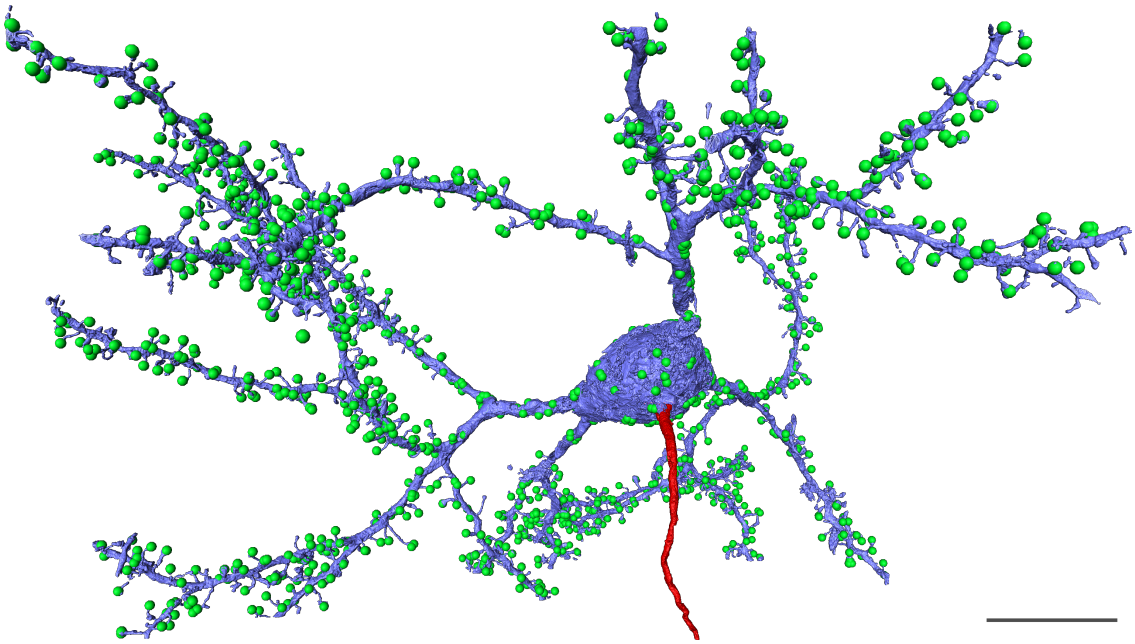

Figure 5.1: Spiny stellate neuron input distribution. Spiny stellate neuron from the SBEM dataset 2012-09-28_ex145_07x2 (Boergens and Helmstaedter, 2012b) from mouse S1 L4 in a volume of size $86 \times 52 \times 86\,\mu\mathrm{m}^3$ (blue isosurface: dendritic tree and soma; red isosurface: axon) shown in the x-y plane. Synapses (green balls) consist of interfaces clustered with a distance threshold of $d = 350\,\mathrm{nm}$. Scale bar: $10\,\mu\mathrm{m}$

spine synapses ($p = 0.14$, one-sample Kolmogorov-Smirnov test, for a clustering threshold of $450\,\mathrm{nm}$) and shaft synapses ($p = 0.55$, one-sample Kolmogorov-Smirnov test, for a clustering threshold of $500\,\mathrm{nm}$).

To examine the distribution of synapses along the dendritic tree of the spiny stellate neuron, the path length from the soma to each synapse along the dendritic tree was calculated for all non-somatic synapses (Figure 5.3). The average distance of spine synapses to the soma was $42.74 \pm 13.72\,\mu\mathrm{m}$ (mean $\pm$ s.d.; range $4.98\,\mu\mathrm{m}$–$87.57\,\mu\mathrm{m}$, $n = 970$, Figure 5.3a, b) while the average distance to the soma for shaft synapses was $29.04 \pm 16.80\,\mu\mathrm{m}$ (mean $\pm$ s.d.; range $1.84\,\mu\mathrm{m}$–$79.51\,\mu\mathrm{m}$, $n = 255$, note that only non-somatic shaft synapses were considered, Figure 5.3a, b) which is significantly different ($p < 10^{-31}$, t-test and Wilcoxon rank sum test). The fraction of shaft synapses among all synapses closer than $40\,\mu\mathrm{m}$ to the soma was 30.0%, while it was only 10% for all synapses with a distance larger than $40\,\mu\mathrm{m}$.

5.4. Discussion

The input map comprising all synapses innervating the dendritic tree of a spiny stellate neuron from a SBEM dataset in mouse S1 L4 was reconstructed using SynEM. In combination with manual annotation of less than 7 hours for a single person, an accuracy of 95% for single synapses was achieved. The distribution of synapse size was found to be well described by a lognormal distribution for all synapses as well as for spine and shaft synapses separately. The average path length from synapses to the soma along the dendrites was found to be smaller for shaft synapses

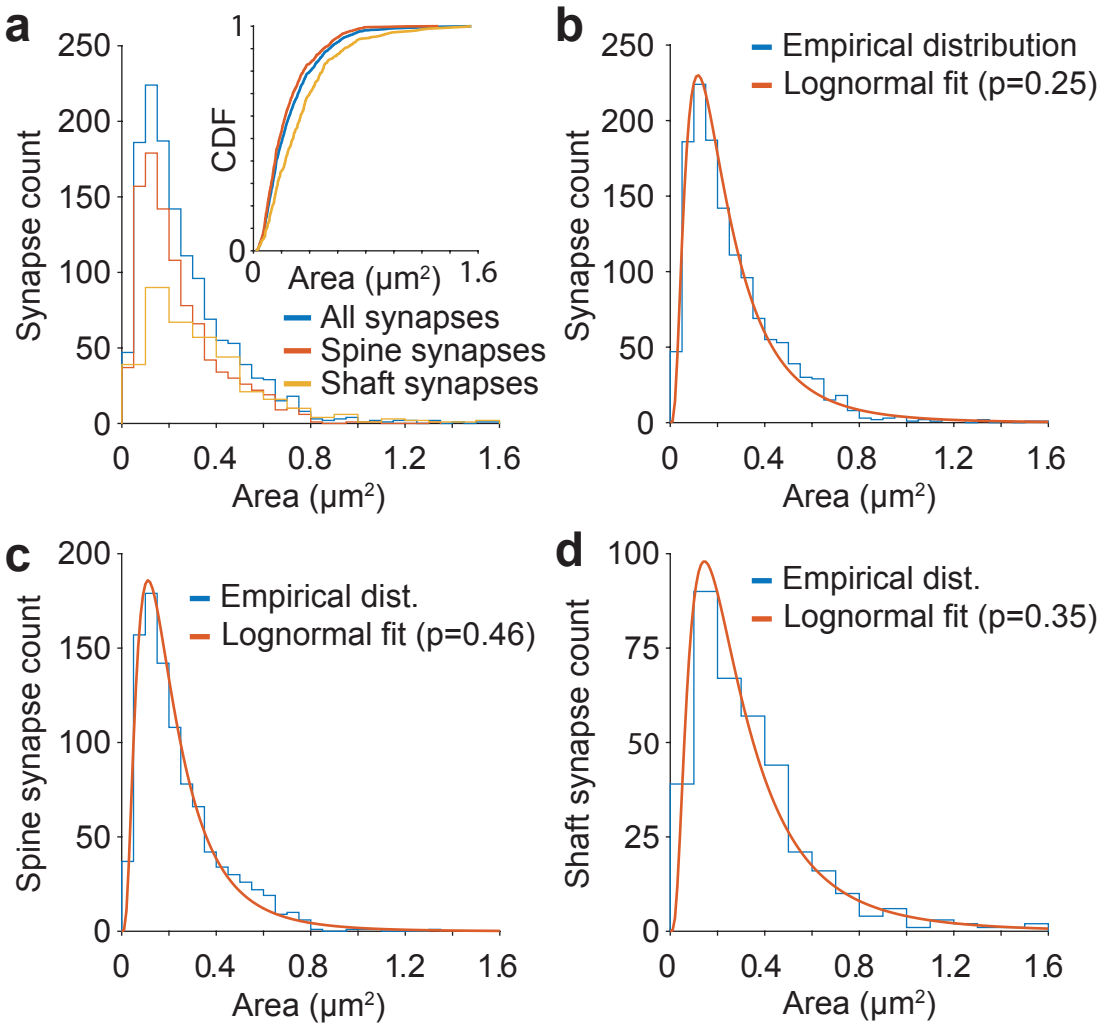

Figure 5.2: Spiny stellate neuron input synapse size distribution. (a) Histogram of synapse area for all synapses (blue) and separately for spine (red) and shaft (yellow) synapses with cumulative density function (CDF) inset at a clustering threshold of $d = 350\,\mathrm{nm}$. (b) Histogram of synapse area for all synapses (blue) with a lognormal fit and the p-value associated to the fit ($p = 0.25$) using a one-sample Kolmogorov-Smirnov test. (c) Same as b but only for spine synapses resulting in a fit with $p = 0.46$ (one-sample Kolmogorov-Smirnov test). (d) Same as b but only for shaft synapses resulting in a fit with $p = 0.35$ (one-sample Kolmogorov-Smirnov test).

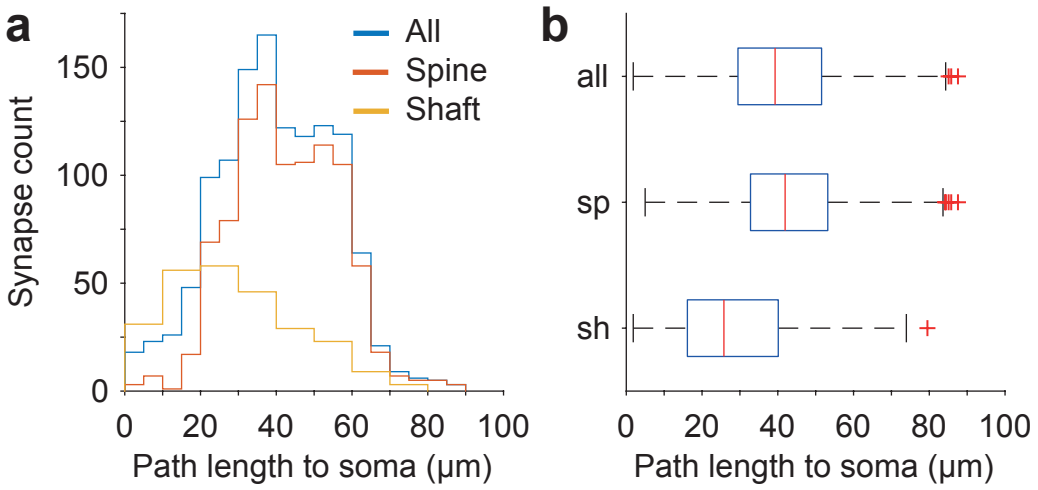

Figure 5.3: Synapse location along the dendritic tree. (a) Distribution of path length from synapse to soma measured from the nodes of the spiny stellate neuron skeleton for all synapses (blue), as well as for spine (red) and shaft (yellow) synapses separately. (b) Box-plot of the path length distributions in (a) for all, spine (sp) and shaft (sh) synapses. For each box (blue) the central mark (red) indicates the median and the left and right edges indicate the 25th and 75th percentiles, respectively. Whiskers extend to smallest and largest points that are not considered as outliers. Outliers are marked with a red plus symbol.

than for spine synapses.

The definition of synapse size used here was the total border area of identified synaptic interfaces. Nearby synaptic interfaces were clustered and their areas were summed up to account for synapses that were split into multiple interfaces. For spine synapses, it was shown (see subsection 3.4.3) that this area measure yields a size distribution that is indistinguishable from the ASI used by de Vivo et al. (2017) assuming that all interfaces of a synapse are detected. For shaft synapses, the total border area of the interface is potentially larger than measures of synapse size involving the PSD or active zone area such as the synaptic apposition surface (SAS) from Santuy et al. (2018), which could result in a heavier tail than expected by a lognormal distribution. Furthermore, due to the larger contact area of shaft synapses typically resulting in a larger number of overlapping interfaces, the clustering procedure is more important and potentially had a stronger effect on the area distribution for shaft synapses than for spine synapse. However, the conclusion that the distribution of synapse size can be fitted by a lognormal distribution was stable for different clustering thresholds.

The finding that shaft synapses are formed to a higher degree near the soma is in accordance with earlier studies for cortical neurons (Anderson et al., 1994) as well as for other brain regions (Megias et al., 2001; Kornfeld et al., 2017), potentially indicating a non-selective veto function by proximal and somatic inhibition (Megias et al., 2001). Note that the analysis of the path length distribution was restricted to the proximal parts of the dendritic tree due to the size of the dataset of $86 \times 52 \times 86\,\mu\mathrm{m}^3$ resulting in a maximal dendrite distance (Euclidean distance) to the soma of at most $40\,\mu\mathrm{m}$ in x- and z-dimension and $20\,\mu\mathrm{m}$ in y direction.

In contrast to previous studies (Anderson et al., 1994; Megias et al., 2001; Kornfeld et al., 2017), synapse types (excitatory, inhibitory) were not identified based on their ultrastructural appearance

but synapses were grouped into spine and shaft synapses. The distinction into spine and shaft synapses is not equivalent to the synapse type (Santuy et al., 2018) potentially resulting in slightly different distributions although the results were consistent.

In conclusion, the combination of SynEM with manual annotation allows answering connectomic questions that previously would have required substantial manual annotation effort. The measured distributions were consistent with previous findings from electrical recordings and EM studies. The population of synapses studied here, namely all synapses that share a common postsynaptic partner, potentially provides an even stronger restriction to models of synaptic plasticity than previously studies synapse populations.

6. Conclusion and Outlook

6.1. Summary

Connectomes - comprehensive maps of neuronal connections of the nervous system of an organism - could help to advance our understanding of the structural as well as the functional principles of the brain. On the nanoscale, this requires to map large volumes of neural tissue at the resolution of single synapses, which is currently only possible using 3D electron microscopy (EM). The reconstruction of a neural network from 3D image data requires to identify all neuronal processes and to detect the synapses between them. Manual reconstruction is prohibitively expensive for all but the smallest neural circuits necessitating the development of automated reconstruction techniques in connectomics. Advances in automated reconstruction of neuronal processes have rendered synapse detection the bottleneck of current connectomic reconstruction approaches. In this thesis, methods for automated synapse detection and volume segmentation for 3D EM-based nanoscale connectomics were developed.

The proposed synapse detection algorithm SynEM formulated synapse detection as binary classification of interfaces between cellular processes from a volume segmentation (chapter 3). The definition of interfaces explicitly included separate subvolumes of the adjacent processes to account for synaptic features extending into the pre- and postsynaptic process. The interface formulation intrinsically includes the partner detection step and allows to naturally incorporate the synapse direction. Interfaces were represented in a feature space using textural and shape descriptors calculated over the interface volumes. SynEM was evaluated on a medium-resolution serial block-face electron microscopy (SBEM) dataset and a high-resolution automated serial section tape-collection scanning electron microscopy (ATUM-SEM) dataset showing superior performance to previously proposed methods. Based on the single synapse prediction performance, the remaining error rates for binary neuron-to-neuron connections were estimated to be less than 3%. The performance of the SynEM interface classification approach was further improved by using texture features learned by convolutional neural networks (CNNs) (chapter 4) while at the same time rendering the need for hand-designed features unnecessary. In addition, several interface classifiers using different feature representations were applied to a novel dataset without retraining or fine-tuning indicating that a simple normalization of the raw data is often enough to get good generalization performance. To improve the segmentation underlying the SynEM interface definition, CNNs were trained to classify membranes between cellular processes followed by a watershed procedure (chapter 4). A novel multi-resolution architecture was proposed showing a substantial improvement over the SegEM baseline (Berning et al., 2015) and to outperform regular CNN architectures and a popular encoder-decoder network architecture (u-net; Ronneberger et al., 2015; Funke et al., 2018). The best segmentation result was achieved using an ensemble of networks. Finally, the SynEM classifier in combination with manual proofreading was used to identify all synapses innervating the dendritic tree of a spiny stellate neuron in a dataset from

mouse primary somatosensory cortex (S1) layer 4 (L4) (chapter 5). The distribution of synapse size, which is correlated to synapse strength, was shown to be well described by a lognormal distribution. Furthermore, the distributions of path lengths from synapses to the soma showed that shaft synapses innervate the spiny stellate more proximal to soma than spine synapses.

6.2. Future Directions

The explicit model of interfaces used by SynEM allows taking all available information about synapse location, orientation and synapse direction into account. The formulation of synapse detection as conditional classification of interfaces provides the additional information contained in a volume segmentation and is thus expected to facilitate the classification task. It is less clear whether other conceptual choices of the SynEM classification approach such as the size and number of subvolumes or the summary statistics are optimal. In the future, more general interface classification setups could be considered that make less explicit assumptions, e.g. by only considering the raw data and binary mask for the involved segments as inputs.

Human annotators often use biological considerations or prior information to resolve difficult or ambiguous locations. These priors can come in the form of hard-constraints that cannot be violated, e.g. that two somata are not connected by neurites or that dendrites are not myelinated, or soft-constraints that make specific configurations more unlikely, e.g. multiple excitatory synapses onto the same spine head. Future models could introduce dependencies between segments and interfaces which allow including various forms of prior information.

6.3. Conclusion

In this thesis, methods for the automated reconstruction of neural circuits from 3D EM data in nanoscale connectomics were presented. The novel synapse detection method SynEM was shown to remove synapse detection as the bottleneck in connectomic reconstruction. The performance of SynEM was further improved by using CNNs to learn features directly from the raw image data. The volume segmentation underlying the SynEM interface definition was improved using a novel multi-resolution architecture. The developed methods were shown to provide substantial advances for the automated reconstruction of 3D EM-based connectomes and, in combination with minimal manual proofreading effort, to be readily usable for biological analyses.

Glossary

A

ANN - artificial neural network 22–24, 61
ASI - axon-spine interface area 50, 51, 92, 97
ATUM-SEM - automated serial section tape-collection scanning electron microscopy 12, 27, 43, 50, 51, 54, 60, 99
AUC - area under curve 25, 26, 67, 74

C

CART - classification and regression trees 20
CNN - convolutional neural network 23–25, 29, 30, 44, 49, 61–65, 67–77, 79, 85, 89, 91, 99, 100

D

DoG - Difference of Gaussians 32, 35

E

ECS - extracellular space 13, 60
EM - electron microscopy 2, 4–6, 8–14, 17, 27–30, 43, 48, 50, 51, 59, 60, 62, 64, 74, 75, 84, 85, 89, 92, 98–100

F

FCN - fully convolutional network 6, 7, 24, 25, 61–64, 74, 77, 85, 89, 91
FFT - fast Fourier transform 16, 79
FIBSEM - focused ion beam scanning electron microscopy 13, 28, 29, 60
FPR - false positive rate 26

I

IED - inter-error distance 62, 80, 81, 84–90

L

L2/3 - layer 2/3 71
L4 - layer 4 ..... 10, 29, 42, 43, 56, 65, 73, 81, 92, 94, 95, 100
LoG - Laplacian of Gaussian 32, 36

M

MAP - maximum a posteriori 20
MLE - maximum likelihood estimate 19, 20, 22
mSEM - multi-beam scanning electron microscopy 13

N

NLL - negative log likelihood 19

P

PSD - postsynaptic density 11, 14, 27–29, 31, 43–45, 47, 59, 60, 71, 74–76, 92, 97

R

ROC - receiver operator characteristic 25, 26

S

S1 - primary somatosensory cortex ... 10, 29, 41–43, 51, 56, 65, 71, 73, 74, 81, 92, 94, 95, 100
SAS - synaptic apposition surface 97
SBEM - serial block-face electron microscopy ... 6, 10, 13, 27, 28, 42–44, 47, 51, 52, 54, 56, 71, 74, 75, 89, 91, 92, 94, 95, 99
SEM - scanning electron microscopy 13
ssTEM - serial section transmission electron microscopy 12, 28, 29

W

wlog - without loss of generality 18

List of Figures

List of Tables

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Anderson, J. C., Douglas, R. J., Martin, K. A. C., and Nelson, J. C. (1994). Map of the synapses formed with the dendrites of spiny stellate neurons of cat visual cortex. *The Journal of Comparative Neurology*, 341(1):25–38.

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916.

Arganda-Carreras, I., Turaga, S. C., Berger, D. R., Cireşan, D., Giusti, A., Gambardella, L. M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J. M., Liu, T., Seyedhosseini, M., Tasdizen, T., Kamentsky, L., Burget, R., Uher, V., Tan, X., Sun, C., Pham, T. D., Bas, E., Uzunbas, M. G., Cardona, A., Schindelin, J., and Seung, H. S. (2015). Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9:142.

Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M., Ferretti, R. E., Leite, R. E., Lent, R., Herculano-Houzel, S., et al. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541.

Barbour, B., Brunel, N., Hakim, V., and Nadal, J.-P. (2007). What can we learn from synaptic weight distributions? *Trends in Neurosciences*, 30(12):622 – 629.

Bargmann, C. I. (2012). Beyond the connectome: how neuromodulators shape neural circuits. *Bioessays*, 34(6):458–465.

Bargmann, C. I. and Marder, E. (2013). From the connectome to brain function. *Nature methods*, 10(6):483–490.

Bartol Jr, T. M., Bromer, C., Kinney, J., Chirillo, M. A., Bourne, J. N., Harris, K. M., and Sejnowski, T. J. (2015). Nanoconnectomic upper bound on the variability of synaptic plasticity. *Elife*, 4:e10778.

Becker, C., Ali, K., Knott, G., and Fua, P. (2013). Learning context cues for synapse segmentation. *IEEE transactions on medical imaging*, 32(10):1864–1877.

Beier, T., Pape, C., Rahaman, N., Prange, T., Berg, S., Bock, D. D., Cardona, A., Knott, G. W., Plaza, S. M., Scheffer, L. K., Koethe, U., Kreshuk, A., and Hamprecht, F. A. (2017). Multicut brings automated neurite segmentation closer to human performance. *Nat Methods*, 14(2):101–102.

Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Berning, M., Boergens, K. M., and Helmstaedter, M. (2015). Segem: efficient image analysis for high-resolution connectomics. *Neuron*, 87(6):1193–1206.

Beucher, S. and Lantuéjoul, C. (1979). Use of watersheds in contour detection. In *In International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation*, volume 132.

Birch-Andersen, A. (1955). Reconstruction of the nuclear sites of salmonella typhimurium from electron micrographs of serial sections. *Microbiology*, 13(2):327–329.

Bishop, C. (2007). Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*.

Boergens, K. M., Berning, M., Bocklisch, T., Bräunlein, D., Drawitsch, F., Frohnhofen, J., Herold, T., Otto, P., Rzepka, N., Werkmeister, T., et al. (2017). webknossos: efficient online 3d data annotation for connectomics. *Nature Methods*.

Boergens, K. M. and Helmstaedter, M. (2012a). 3D SBEM Dataset: Primary somatosensory cortex layer 2/3 of P28 C57BL/6 male mouse (size: $97 \times 61 \times 196\,\mu\mathrm{m}^3$, voxel size: $12 \times 12 \times 26\,\mathrm{nm}^3$), 2017-11-16_ex144_st08x2. Unpublished data.

Boergens, K. M. and Helmstaedter, M. (2012b). 3D SBEM Dataset: Primary somatosensory cortex layer 4 of P28 C57BL/6 male mouse (size: $93 \times 60 \times 93\,\mu\mathrm{m}^3$, voxel size: $11.24 \times 11.24 \times 28\,\mathrm{nm}^3$), 2012-09-28_ex145. Partly published: `https://segem.rzg.mpg.de/`, `https://demo.webknossos.org/datasets/2012-09-28_ex145_07x2_demo/view` and `http://synem.rzg.mpg.de/`.

Boergens, K. M. and Helmstaedter, M. (2012c). 3D SBEM Dataset: Primary somatosensory cortex layer 4 of P28 C57BL/6 male mouse (size: $96 \times 64 \times 96\,\mu\mathrm{m}^3$, voxel size: $11.24 \times 11.24 \times 28\,\mathrm{nm}^3$), 2012-09-28_ex145_ROI2017. Unpublished data.

Bogovic, J. A., Huang, G. B., and Jain, V. (2013). Learned versus hand-designed feature representations for 3d agglomeration. *arXiv preprint arXiv:1312.6159*.

Braitenberg, V. and Schuez, A. (1991). *Anatomy of The Cortex. Statistics and Geometry*. Berlin: Springer-Verlag.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. Wadsworth Advanced Books and Software, Belmont, CA.

Briggman, K. L. and Bock, D. D. (2012). Volume electron microscopy for neuronal circuit reconstruction. *Current opinion in neurobiology*, 22(1):154–161.

Briggman, K. L., Helmstaedter, M., and Denk, W. (2011). Wiring specificity in the direction-selectivity circuit of the retina. *Nature*, 471(7337):183–188.

Bucher, D. and Marder, E. (2013). Snapshot: Neuromodulation. *Cell*, 155(2):482–482 e1.

Buhmann, J., Krause, R., Lentini, R. C., Eckstein, N., Cook, M., Turaga, S., and Funke, J. (2018). Synaptic partner prediction from point annotations in insect brains. *arXiv preprint arXiv:1806.08205*.

Buzsaki, G. and Mizuseki, K. (2014). The log-dynamic brain: how skewed distributions affect network operations. *Nat Rev Neurosci*, 15(4):264–78.

Cheetham, C. E., Barnes, S. J., Albieri, G., Knott, G. W., and Finnerty, G. T. (2012). Pansynaptic enlargement at adult cortical connections strengthened by experience. *Cerebral cortex*, 24(2):521–531.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848.

Ciresan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc.

Colonnier, M. (1968). Synaptic patterns on different cell types in the different laminae of the cat visual cortex. an electron microscope study. *Brain Res*, 9(2):268–87.

de Vivo, L., Bellesi, M., Marshall, W., Bushong, E. A., Ellisman, M. H., Tononi, G., and Cirelli, C. (2017). Ultrastructural evidence for synaptic scaling across the wake/sleep cycle. *Science*, 355(6324):507–510.

Denk, W., Briggman, K. L., and Helmstaedter, M. (2012). Structural neurobiology: missing link to a mechanistic understanding of neural computation. *Nature Reviews Neuroscience*, 13(5):351–358.

Denk, W. and Horstmann, H. (2004). Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS biology*, 2(11):e329.

Dorkenwald, S., Schubert, P. J., Killinger, M. F., Urban, G., Mikula, S., Svara, F., and Kornfeld, J. (2017). Automated synaptic connectivity inference for volume electron microscopy. *Nature Methods*, 14(4):435–442.

Drozdzal, M., Chartrand, G., Vorontsov, E., Shakeri, M., Jorio, L. D., Tang, A., Romero, A., Bengio, Y., Pal, C., and Kadoury, S. (2018). Learning normalized inputs for iterative estimation in medical image segmentation. *Medical Image Analysis*, 44:1 – 13.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Eberle, A., Mikula, S., Schalek, R., Lichtman, J., Tate, M. K., and Zeidler, D. (2015). High-resolution, high-throughput imaging with a multibeam scanning electron microscope. *Journal of microscopy*, 259(2):114–120.

Feldmeyer, D., Egger, V., Lubke, J., and Sakmann, B. (1999). Reliable synaptic connections between pairs of excitatory layer 4 neurones within a single 'barrel' of developing rat somatosensory cortex. *J Physiol*, 521 Pt 1:169–90.

Feldmeyer, D., Lubke, J., and Sakmann, B. (2006). Efficacy and connectivity of intracolumnar pairs of layer 2/3 pyramidal cells in the barrel cortex of juvenile rats. *J Physiol*, 575(Pt 2):583–602.

Feldmeyer, D., Lubke, J., Silver, R. A., and Sakmann, B. (2002). Synaptic connections between layer 4 spiny neurone-layer 2/3 pyramidal cell pairs in juvenile rat barrel cortex: physiology and anatomy of interlaminar signalling within a cortical column. *J Physiol*, 538(Pt 3):803–22.

Fornito, A., Zalesky, A., and Breakspear, M. (2015). The connectomics of brain disorders. *Nat Rev Neurosci*, 16(3):159–72.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Frick, A., Feldmeyer, D., Helmstaedter, M., and Sakmann, B. (2008). Monosynaptic connections between pairs of l5a pyramidal neurons in columns of juvenile rat somatosensory cortex. *Cereb Cortex*, 18(2):397–406.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–374.

Funke, J., Klein, J., Moreno-Noguer, F., Cardona, A., and Cook, M. (2017). Ted: A tolerant edit distance for segmentation evaluation. *Methods*, 115:119 – 127. Image Processing for Biologists.

Funke, J., Tschopp, F. D., Grisaitis, W., Sheridan, A., Singh, C., Saalfeld, S., and Turaga, S. C. (2018). Large scale image segmentation with structured loss based deep learning for connectome reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2016). Image style transfer using convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423.

Gibson, J. R., Beierlein, M., and Connors, B. W. (1999). Two networks of electrically coupled inhibitory neurons in neocortex. *Nature*, 402(6757):75–9.

Giusti, A., Ciresan, D. C., Masci, J., Gambardella, L. M., and Jürgen Schmidhuber (2013). Fast image scanning with deep max-pooling convolutional neural networks. *2013 IEEE International Conference on Image Processing*, pages 4034–4038.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Golgi, C. (1873). Sulla struttura della sostanza grigia del cervello. *Gazzetta Medica Italiana. Lombardia*, 33:244–246.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *NIPS*.

Grais, E. M., Wierstorf, H., Ward, D., and Plumbley, M. D. (2017). Multi-resolution fully convolutional neural networks for monaural audio source separation. *CoRR*, abs/1710.11473.

Gray, E. G. (1959). Axo-somatic and axo-dendritic synapses of the cerebral cortex: an electron microscope study. *Journal of anatomy*, 93(Pt 4):420.

Gupta, A., Wang, Y., and Markram, H. (2000). Organizing principles for a diversity of gabaergic interneurons and synapses in the neocortex. *Science*, 287(5451):273–8.

Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12:993–1001.

Harris, K. M., Perry, E., Bourne, J., Feinberg, M., Ostroff, L., and Hurlburt, J. (2006). Uniform serial sectioning for transmission electron microscopy. *Journal of Neuroscience*, 26(47):12101–12103.

Harris, K. M. and Stevens, J. K. (1989). Dendritic spines of ca 1 pyramidal cells in the rat hippocampus: serial electron microscopy with reference to their biophysical characteristics. *Journal of Neuroscience*, 9(8):2982–2997.

Harris, K. M. and Weinberg, R. J. (2012). Ultrastructure of synapses in the mammalian brain. *Cold Spring Harbor perspectives in biology*, 4(5):a005587.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. M. (2017). Neuroscience-inspired artificial intelligence. *Neuron*, 95 2:245–258.

Hayworth, K., Kasthuri, N., Schalek, R., and Lichtman, J. (2006). Automating the collection of ultrathin serial sections for large volume tem reconstructions. *Microscopy and Microanalysis*, 12(S02):86.

Hayworth, K. J., Xu, C. S., Lu, Z., Knott, G. W., Fetter, R. D., Tapia, J. C., Lichtman, J. W., and Hess, H. F. (2015). Ultrastructurally smooth thick partitioning and volume stitching for large-scale connectomics. *Nature methods*, 12(4):319–322.

He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 346–361, Cham. Springer International Publishing.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hebb, D. O. (1949). The organization of behavior: A neurophysiological approach.

Heinrich, L., Funke, J., Pape, C., Nunez-Iglesias, J., and Saalfeld, S. (2018). Synaptic cleft segmentation in non-isotropic volume electron microscopy of the complete drosophila brain. *arXiv preprint arXiv:1805.02718*.

Helmstaedter, M. (2013). Cellular-resolution connectomics: challenges of dense neural circuit reconstruction. *Nature methods*, 10(6):501–507.

Helmstaedter, M. (2015). The mutual inspirations of machine learning and neuroscience. *Neuron*, 86(1):25–8.

Helmstaedter, M., Briggman, K. L., and Denk, W. (2011). High-accuracy neurite reconstruction for high-throughput neuroanatomy. *Nature neuroscience*, 14(8):1081–1088.

Helmstaedter, M., Briggman, K. L., Turaga, S. C., Jain, V., Seung, H. S., and Denk, W. (2013). Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500(7461):168–174.

Hoffmann, J. H., Meyer, H. S., Schmitt, A. C., Straehle, J., Weitbrecht, T., Sakmann, B., and Helmstaedter, M. (2015). Synaptic conductance estimates of the connection between local inhibitor interneurons and pyramidal neurons in layer 2/3 of a cortical column. *Cereb Cortex*, 25(11):4415–29.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

Huang, G. B. and Plaza, S. (2014). Identifying synapses using deep and wide multiscale recursive networks. *arXiv preprint arXiv:1409.1789*.

Huang, G. B., Scheffer, L. K., and Plaza, S. M. (2016). Fully-automatic synapse prediction and validation on a large data set. *arXiv preprint arXiv:1604.03075*.

Hyafil, L. and Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15 – 17.

Jagadeesh, V., Anderson, J., Jones, B., Marc, R., Fisher, S., and Manjunath, B. (2014). Synapse classification and localization in electron micrographs. *Pattern Recognition Letters*, 43:17–24.

Jain, V. (2017). Adversarial image alignment and interpolation. *arXiv preprint arXiv:1707.00067*.

Jain, V., Bollmann, B., Richardson, M., Berger, D. R., Helmstaedter, M. N., Briggman, K. L., Denk, W., Bowden, J. B., Mendenhall, J. M., Abraham, W. C., Harris, K. M., Kasthuri, N., Hayworth, K. J., Schalek, R., Tapia, J. C., Lichtman, J. W., and Seung, H. S. (2010a). Boundary learning by optimization with topological constraints. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2488–2495.

Jain, V., Murray, J. F., Roth, F., Turaga, S., Zhigulin, V., Briggman, K. L., Helmstaedter, M. N., Denk, W., and Seung, H. S. (2007). Supervised learning of image restoration with convolutional networks. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.

Jain, V., Seung, H. S., and Turaga, S. C. (2010b). Machines that learn to segment images: a crucial technology for connectomics. *Current opinion in neurobiology*, 20(5):653–666.

Januszewski, M., Kornfeld, J., Li, P. H., Pope, A., Blakely, T., Lindsey, L., Maitin-Shepard, J. B., Tyka, M., Denk, W., and Jain, V. (2017). High-precision automated reconstruction of neurons with flood-filling networks. *bioRxiv*, page 200675.

Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., Hudspeth, A. J., et al. (2000). *Principles of neural science*, volume 4. McGraw-hill New York.

Kasthuri, N., Hayworth, K. J., Berger, D. R., Schalek, R. L., Conchello, J. A., Knowles-Barley, S., Lee, D., Vázquez-Reina, A., Kaynig, V., Jones, T. R., et al. (2015). Saturated reconstruction of a volume of neocortex. *Cell*, 162(3):648–661.

Kaynig, V., Fischer, B., Müller, E., and Buhmann, J. M. (2010). Fully automatic stitching and distortion correction of transmission electron microscope images. *Journal of structural biology*, 171(2):163–173.

Ke, M.-T., Nakai, Y., Fujimoto, S., Takayama, R., Yoshida, S., Kitajima, T. S., Sato, M., and Imai, T. (2016). Super-resolution mapping of neuronal circuitry with an index-optimized clearing agent. *Cell reports*, 14(11):2718–2732.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Knoll, M. and Ruska, E. (1932). Das elektronenmikroskop. *Zeitschrift für Physik A Hadrons and Nuclei*, 78(5):318–339.

Knott, G., Marchman, H., Wall, D., and Lich, B. (2008). Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. *Journal of Neuroscience*, 28(12):2959–2964.

Koelbl, C., Helmstaedter, M., Lubke, J., and Feldmeyer, D. (2015). A barrel-related interneuron in layer 4 of rat somatosensory cortex with a high intrabarrel connectivity. *Cereb Cortex*, 25(3):713–25.

Kornfeld, J., Benezra, S. E., Narayanan, R. T., Svara, F., Egger, R., Oberlaender, M., Denk, W., and Long, M. A. (2017). Em connectomics reveals axonal target variation in a sequence-generating network. *eLife*, 6:e24364.

Kornfeld, J. and Denk, W. (2018). Progress and remaining challenges in high-throughput volume electron microscopy. *Current Opinion in Neurobiology*, 50:261 – 267. Neurotechnologies.

Korogod, N., Petersen, C. C., and Knott, G. W. (2015). Ultrastructural analysis of adult mouse neocortex comparing aldehyde perfusion with cryo fixation. *Elife*, 4.

Kreshuk, A., Funke, J., Cardona, A., and Hamprecht, F. A. (2015). Who is talking to whom: synaptic partner detection in anisotropic volumes of insect brain. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 661–668. Springer.

Kreshuk, A., Koethe, U., Pax, E., Bock, D. D., and Hamprecht, F. A. (2014). Automated detection of synapses in serial section transmission electron microscopy image stacks. *PloS one*, 9(2):e87351.

Kreshuk, A., Straehle, C. N., Sommer, C., Koethe, U., Cantoni, M., Knott, G., and Hamprecht, F. A. (2011). Automated detection and segmentation of synaptic contacts in nearly isotropic serial electron microscopy images. *PloS one*, 6(10):e24899.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Kroeger, T., Mikula, S., Denk, W., Koethe, U., and Hamprecht, F. A. (2013). Learning to segment neurons with non-local quality measures. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 419–427. Springer.

Kubota, Y., Karube, F., Nomura, M., and Kawaguchi, Y. (2016). The diversity of cortical inhibitory synapses. *Frontiers in Neural Circuits*, 10:27.

Lakadamyali, M., Babcock, H., Bates, M., Zhuang, X., and Lichtman, J. (2012). 3d multicolor super-resolution imaging offers improved accuracy in neuron tracing. *PLOS ONE*, 7(1):1–10.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. (2017). Building machines that learn and think like people. *The Behavioral and brain sciences*, 40:e253.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer.

Lee, K., Zung, J., Li, P., Jain, V., and Seung, H. S. (2017). Superhuman accuracy on the snemi3d connectomics challenge. *arXiv preprint arXiv:1706.00120*.

Lichtman, J. W. and Denk, W. (2011). The big and the small: challenges of imaging the brain's circuits. *Science*, 334(6056):618–623.

Lichtman, J. W., Pfister, H., and Shavit, N. (2014). The big data challenges of connectomics. *Nat Neurosci*, 17(11):1448–54.

Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

Lynch, M. A. (2004). Long-term potentiation and memory. *Physiological Reviews*, 84(1):87–136. PMID: 14715912.

Mallat, S. (1999). *A wavelet tour of signal processing*. Academic press.

Marder, E. (2012). Neuromodulation of neuronal circuits: back to the future. *Neuron*, 76(1):1–11.

Markram, H., Lubke, J., Frotscher, M., Roth, A., and Sakmann, B. (1997). Physiology and anatomy of synaptic connections between thick tufted pyramidal neurones in the developing rat neocortex. *J Physiol*, 500 (Pt 2):409–40.

Markram, H., Toledo-Rodriguez, M., Wang, Y., Gupta, A., Silberberg, G., and Wu, C. (2004). Interneurons of the neocortical inhibitory system. *Nat Rev Neurosci*, 5(10):793–807.

Masci, J., Giusti, A., Ciresan, D. C., Fricout, G., and Jürgen Schmidhuber (2013). A fast learning algorithm for image segmentation with max-pooling convolutional networks. *2013 IEEE International Conference on Image Processing*, pages 2713–2717.

Megias, M., Emri, Z., Freund, T. F., and Gulyas, A. I. (2001). Total number and distribution of inhibitory and excitatory synapses on hippocampal ca1 pyramidal cells. *Neuroscience*, 102(3):527–40.

Merchán-Pérez, A., Rodríguez, J.-R., González, S., Robles, V., DeFelipe, J., Larrañaga, P., and Bielza, C. (2014). Three-dimensional spatial distribution of synapses in the neocortex: A dual-beam electron microscopy study. *Cerebral Cortex*, 24(6):1579–1588.

Mishchenko, Y., Hu, T., Spacek, J., Mendenhall, J., Harris, K. M., and Chklovskii, D. B. (2010). Ultrastructural analysis of hippocampal neuropil from the connectomics perspective. *Neuron*, 67(6):1009–1020.

Morgan, J. L. and Lichtman, J. W. (2013). Why not connectomics? *Nature methods*, 10(6):494–500.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Murre, J. and Sturdy, D. P. (1995). The connectivity of the brain: multi-level quantitative analysis. *Biological cybernetics*, 73(6):529–545.

Navlakha, S., Suhan, J., Barth, A. L., and Bar-Joseph, Z. (2013). A high-throughput framework to detect synapses in electron microscopy images. *Bioinformatics*, 29(13):i9–i17.

Neila, P. M., Baumela, L., González-Soriano, J., Rodríguez, J.-R., DeFelipe, J., and Merchán-Pérez, A. (2016). A fast method for the segmentation of synaptic junctions and mitochondria in serial electron microscopic images of the brain. *Neuroinformatics*, 14(2):235–250.

Oh, S. W., Harris, J. A., Ng, L., Winslow, B., Cain, N., Mihalas, S., Wang, Q., Lau, C., Kuan, L., Henry, A. M., et al. (2014). A mesoscale connectome of the mouse brain. *Nature*, 508(7495):207–214.

Pal, N. R. and Pal, S. K. (1993). A review on image segmentation techniques. *Pattern Recognition*, 26:1277–1294.

Palay, S. L. (1956). Synapses in the central nervous system. *The Journal of Cell Biology*, 2(4):193–202.

Palay, S. L. and Palade, G. E. (1955). The fine structure of neurons. *The Journal of biophysical and biochemical cytology*, 1(1):69.

Pallotto, M., Watkins, P. V., Fubara, B., Singer, J. H., and Briggman, K. L. (2015). Extracellular space preservation aids the connectomic analysis of neural circuits. *Elife*, 4.

Parag, T., Berger, D., Kamentsky, L., Staffler, B., Wei, D., Helmstaedter, M., Lichtman, J. W., and Pfister, H. (2018). Detecting synapse location and connectivity by signed proximity estimation and pruning with deep nets. *arXiv preprint arXiv:1807.02739*.

Pelt, D. M. and Sethian, J. A. (2018). A mixed-scale dense convolutional neural network for image analysis. *Proceedings of the National Academy of Sciences*, 115(2):254–259.

Perea, G., Sur, M., and Araque, A. (2014). Neuron-glia networks: integral gear of brain function. *Frontiers in Cellular Neuroscience*, 8:378.

Perez, A. J., Seyedhosseini, M., Deerinck, T. J., Bushong, E. A., Panda, S., Tasdizen, T., and Ellisman, M. H. (2014). A workflow for the automatic segmentation of organelles in electron microscopy image stacks. *Frontiers in neuroanatomy*, 8.

Peters, A. and Palay, S. L. (1996). The morphology of synapses. *J Neurocytol*, 25(12):687–700.

Petilla Interneuron Nomenclature Group, G., Ascoli, G. A., Alonso-Nanclares, L., Anderson, S. A., Barrionuevo, G., Benavides-Piccione, R., Burkhalter, A., Buzsaki, G., Cauli, B., Defelipe, J., Fairen, A., Feldmeyer, D., Fishell, G., Fregnac, Y., Freund, T. F., Gardner, D., Gardner, E. P., Goldberg, J. H., Helmstaedter, M., Hestrin, S., Karube, F., Kisvarday, Z. F., Lambolez, B., Lewis, D. A., Marin, O., Markram, H., Munoz, A., Packer, A., Petersen, C. C., Rockland, K. S., Rossier, J., Rudy, B., Somogyi, P., Staiger, J. F., Tamas, G., Thomson, A. M., Toledo-Rodriguez, M., Wang, Y., West, D. C., and Yuste, R. (2008). Petilla terminology: nomenclature of features of gabaergic interneurons of the cerebral cortex. *Nat Rev Neurosci*, 9(7):557–68.

Pham, D. L., Xu, C., and Prince, J. (2000). Current methods in medical image segmentation. *Annual review of biomedical engineering*, 2:315–37.

Plaza, S. M., Parag, T., Huang, G. B., Olbris, D. J., Saunders, M. A., and Rivlin, P. K. (2014). Annotating synapses in large em datasets. *arXiv preprint arXiv:1409.1801*.

Rall, L. B. (1981). *Automatic differentiation: Techniques and applications*. Springer Verlag, Berlin.

Rees, C. L., Moradi, K., and Ascoli, G. A. (2017). Weighing the evidence in peters' rule: Does neuronal morphology predict connectivity? *Trends Neurosci*, 40(2):63–71.

Revel, J. P. and Karnovsky, M. J. (1967). Hexagonal array of subunits in intercellular junctions of the mouse heart and liver. *J Cell Biol*, 33(3):C7–C12.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

Robertson, J. D. (1953). Ultrastructure of two invertebrate synapses. *Proceedings of the Society for Experimental Biology and Medicine*, 82(2):219–223.

Robertson, J. D. (1963). The occurrence of a subunit pattern in the unit membranes of club endings in mauthner cell synapses in goldfish brains. *The Journal of Cell Biology*, 19(1):201–221.

Rolnick, D., Meirovitch, Y., Parag, T., Pfister, H., Jain, V., Lichtman, J. W., Boyden, E. S., and Shavit, N. (2017). Morphological error detection in 3d segmentations. *arXiv preprint arXiv:1705.10882*.

Roncal, W. G., Pekala, M., Kaynig-Fittkau, V., Kleissas, D. M., Vogelstein, J. T., Pfister, H., Burns, R., Vogelstein, R. J., Chevillet, M. A., and Hager, G. D. (2015). Vesicle: Volumetric evaluation of synaptic interfaces using computer vision at large scale. In Xie, X., Jones, M. W., and Tam, G. K. L., editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 81.1–81.13. BMVA Press.

Ronneberger, O., P.Fischer, and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer. (available on arXiv:1505.04597 [cs.CV]).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Rust, M. J., Bates, M., and Zhuang, X. (2006). Sub-diffraction-limit imaging by stochastic optical reconstruction microscopy (storm). *Nature Methods*, 3:793.

Saalfeld, S., Cardona, A., Hartenstein, V., and Tomancak, P. (2009). Catmaid: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics*, 25(15):1984–6.

Saalfeld, S., Fetter, R., Cardona, A., and Tomancak, P. (2012). Elastic volume reconstruction from series of ultra-thin microscopy sections. *Nature methods*, 9(7):717–720.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.

Santuy, A., Rodríguez, J.-R., DeFelipe, J., and Merchán-Pérez, A. (2018). Study of the size and shape of synapses in the juvenile rat somatosensory cortex with 3d electron microscopy. *eNeuro*, 5(1).

Schikorski, T. and Stevens, C. F. (1997). Quantitative ultrastructural analysis of hippocampal excitatory synapses. *Journal of Neuroscience*, 17(15):5858–5867.

Schmidt, H., Gour, A., Straehle, J., Boergens, K. M., Brecht, M., and Helmstaedter, M. (2017). Axonal synapse sorting in medial entorhinal cortex. *Nature*, 549(7673):469–475.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. *International Conference on Learning Representations (ICLR)*.

Seung, H. S. (2009). Reading the book of memory: sparse sampling versus dense mapping of connectomes. *Neuron*, 62(1):17–29.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Sommer, C., Straehle, C., Kothe, U., and Hamprecht, F. A. (2011). Ilastik: Interactive learning and segmentation toolkit. *2011 8th Ieee International Symposium on Biomedical Imaging: From Nano to Macro*, pages 230–233.

Song, S., Sjöström, P. J., Reigl, M., Nelson, S., and Chklovskii, D. B. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLOS Biology*, 3(3).

Sporns, O., Tononi, G., and Kötter, R. (2005). The human connectome: A structural description of the human brain. *PLOS Computational Biology*, 1(4).

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Staffler, B., Berning, M., Boergens, K. M., Gour, A., Smagt, P. v. d., and Helmstaedter, M. (2017a). Synem, automated synapse detection for connectomics. *eLife*, 6:e26414.

Staffler, B., Berning, M., Boergens, K. M., Gour, A., van der Smagt, P., and Helmstaedter, M. (2017b). Synem: Automated synapse detection for connectomics. *bioRxiv*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Takemura, S. Y., Bharioke, A., Lu, Z., Nern, A., Vitaladevuni, S., Rivlin, P. K., Katz, W. T., Olbris, D. J., Plaza, S. M., Winston, P., Zhao, T., Horne, J. A., Fetter, R. D., Takemura, S., Blazek, K., Chang, L. A., Ogundeyi, O., Saunders, M. A., Shapiro, V., Sigmund, C., Rubin, G. M., Scheffer, L. K., Meinertzhagen, I. A., and Chklovskii, D. B. (2013). A visual motion detection circuit suggested by drosophila connectomics. *Nature*, 500(7461):175–81.

Turaga, S. C., Briggman, K. L., Helmstaedter, M., Denk, W., and Seung, H. S. (2009). Maximin affinity learning of image segmentation. In *NIPS*.

Turaga, S. C., Murray, J. F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W., and Seung, H. S. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural computation*, 22(2):511–538.

Unnikrishnan, R., Pantofaru, C., and Hebert, M. (2007). Toward objective evaluation of image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):929–944.

Vizi, E. S. (2000). Role of high-affinity receptors and membrane transporters in nonsynaptic communication and drug action in the central nervous system. *Pharmacol Rev*, 52(1):63–89.

Wanner, A., Kirschmann, M., and Genoud, C. (2015). Challenges of microtome-based serial block-face scanning electron microscopy in neuroscience. *Journal of microscopy*, 259(2):137–142.

White, E. L. and Peters, A. (1993). Cortical modules in the posteromedial barrel subfield (sml) of the mouse. *Journal of Comparative Neurology*, 334(1):86–96.

White, J. G., Southgate, E., Thomson, J. N., and Brenner, S. (1986). The structure of the nervous system of the nematode caenorhabditis elegans: the mind of a worm. *Phil. Trans. R. Soc. Lond*, 314:1–340.

Whitlock, J. R., Heynen, A. J., Shuler, M. G., and Bear, M. F. (2006). Learning induces long-term potentiation in the hippocampus. *Science*, 313(5790):1093–1097.

Wolf, S., Schott, L., and Köthe and Fred A. Hamprecht, U. (2017). Learned watershed: End-to-end learning of seeded segmentation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2030–2038.

y Cajal, S. R. (1888). *Estructura de los centros nerviosos de las aves*. Jiménez y Molina.

Yoo, I., Hildebrand, D. G. C., Tobin, W. F., Lee, W.-C. A., and Jeong, W.-K. (2017). ssemnet: Serial-section electron microscopy image registration using a spatial transformer network with learned features. In *DLMIA/ML-CDS@MICCAI*.

Zlateski, A., Lee, K., and Seung, H. S. (2016). Znn – a fast and scalable algorithm for training 3d convolutional networks on multi-core and many-core shared memory machines. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 801–811.

Zung, J., Tartavull, I., Lee, K., and Seung, H. S. (2017). An error detection and correction framework for connectomics. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6821–6832. Curran Associates, Inc.