Methods for the acquisition and analysis of volume electron microscopy data

Philipp Johannes Schubert

München 2022

Methods for the acquisition and analysis of volume electron microscopy data

Philipp Johannes Schubert

Dissertation
der Fakultät für Physik
der Ludwig-Maximilians-Universität
München

vorgelegt von
Philipp Johannes Schubert
aus Wiesbaden

München, den 17.02.2022

Erstgutachter: Prof. Dr. Winfried Denk

Zweitgutachter: Prof. Dr. Daniel Rückert

Tag der mündlichen Prüfung: 05.04.2022

Contents

List of Figures

List of Tables

Zusammenfassung

Technologische Fortschritte in den Bereichen der Elektronenmikroskopie (EM) und Computerhardware haben es ermöglicht die Erfassung und Analyse von elektronenmikroskopischen Volumendaten zu beschleunigen. Mit der Fähigkeit synaptische Schaltkreise mit hohem Durchsatz und hoher Genauigkeit zu rekonstruieren, können Fragen beantwortet werden, die große Probenvolumen oder verschiedene Konditionen voraussetzen, wie zum Beispiel das Lernen des Gesangs bei männlichen Zebrafinken. Die hochauflösenden Bilddaten ermöglichen Einblicke in Strukturen die kleiner sind als einzelne Synapsen sowie eine detailierte morphologische Rekonstruktion von Neuronen, erfordern aber ein hohes Maß an Automatisierung um die immer größeren Volumen analysieren zu können. Die Automatisierung ist letztendlich von der Vorhersagegenauigkeit der zugrundeliegenden Modelle abhängig, welche wiederum von der Bildqualität beeinflusst wird. Serielle Rasterelektronenmikroskopie von Blockoberflächen einzelner Proben weist weniger Bildfehler auf als Ansätze, die auf ultra-dünnen Serienschnitten basieren, hat aber einen begrenzten Durchsatz bei der Bildaufnahme. Diese Dissertation umfasst Experimente und Methoden zur Aufnahme und zur automatisierten Datenanalyse von volumetrischen EM-Bilddaten.

Das erste Kapitel behandelt ein bildgebendes Verfahren mittels Elektronenmikroskopie von Blockoberflächen, welches in einem Durchgang die Oberfläche von mehreren dicken Probenschnitten mit Hilfe eines Gascluster-Ionenstrahls (engl. gas cluster ion beam, GCIB) abträgt. Es ist unablässlich eine gute Bildqualität zu gewährleisten, insbesondere bei Techniken, die Blockoberflächen unwiederbringlich entfernen und eine nachträgliche Neuaufnahme ausschließen, was im Mindesten einen scharfen Fokus voraussetzt. Zu diesem Zweck wurde eine schnelle und zuverlässige Routine für die Korrektur der Fokusparameter basierend auf tiefen künstlichen neuronalen Netzwerken entwickelt, die auch bei niedrigen Signal-Rausch-Verhältnissen anwendbar ist.

In den letzten beiden Kapiteln werden Methoden zur Automatisierung der Analyseschritte eingeführt. Als eine Alternative zur semantischen Segmentierung dichter Voxeldaten werden zwei Modelle vorgestellt, die auf tiefen künstlichen neuronalen Netzwerken beruhen und auf den Oberflächenreprseäntationen von segmentierten Zellen arbeiten. Deren Anwendbarkeit wird an den Problemen der semantischen Segmentierung kleiner Zellkompartimente, beispielsweise Köpfe von Dornenfortsätzen (engl. spines), Zelltypenklassifizierung und der unüberwachten Clusteranalyse von Neuronen demonstriert. Diese Modelle werden mit der Identifikation von zeullulärer Ultrastruktur und Synapsenrekonstruktion in einem Konnektom-Analyse Softwarepaket (*SyConn2*) zusammengeführt, welches auf die Nutzung von Hochleistungsrechnern optimiert und auf Konnektomdatensätzen von bis zu zehn Teravoxeln getestet wurde.

Abstract

Technological advances in electron microscopy (EM) imaging and computer hardware have made it possible to accelerate the acquisition and analysis of volume electron microscope data. With the capability to extract synaptic wiring diagrams at both high throughput and accuracy, questions can be addressed that, for instance, require large or multiple tissue samples under varying conditions, such as song learning in male zebra finches. The high-resolution image data allows inspection of structures smaller than individual synapses and a detailed morphological reconstruction of neurons but demands a substantial degree of automation to analyze increasingly large volumes. Automation ultimately depends on the achieved accuracy, which in turn is affected by the image quality. Serial block-face scanning electron microscopy uses a single specimen block and has fewer image defects than approaches based on serial ultrathin sectioning but is limited in throughput during image acquisition. This dissertation covers experiments and methods for the acquisition and automated data analysis of volumetric EM image data.

The first chapter outlines an imaging pipeline with block-face scanning electron microscopy, which uses a gas cluster ion beam (GCIB) to ablate the surface of multiple thick specimen sections in one pass. Maintaining good image quality throughout the acquisition is crucial, in particular for destructive block-face techniques that make subsequent re-imaging impossible, which at the very least requires sharp focus. To this end, a fast and robust routine for focus parameter correction based on deep artificial neural networks was developed that is applicable under low signal-to-noise conditions.

The last two chapters introduce methods for automating the analysis steps. Two deep learning models are introduced as an alternative to the semantic segmentation of dense voxel data, which operate on the surface reconstruction of segmented cells. Their application is demonstrated on the tasks of semantic segmentation of fine cell compartments, such as spine heads, cell type classification, and unsupervised clustering of neurons. These models are integrated with the identification of cellular ultrastructure and synapse reconstruction in a connectome analysis framework (*SyConn2*), which was optimized to use high-performance compute environments and tested on connectomic data sets of up to ten teravoxels.

Chapter 1

Introduction

1.1 Connectomics

The structural study of neurons originated in the second half of the 19th century with Santiago Ramón y Cajal, who used light microscopes and a sparse staining technique to capture the detailed anatomy of cells in drawings (Ramón y Cajal 1888). His pioneering work led to the concept that the nervous system is built from individual, interconnected cells.

With the invention of the transmission electron microscope (TEM) (Knoll and Ruska 1932) and scanning electron microscope (SEM) (von Ardenne 1938) in the 1930s, it was possible to study neural tissue at a much higher resolution compared to light microscopy. In contrast to the sparse staining used by Ramón y Cajal, electron microscopes allowed the visualization of neurons with dense staining. After more than a decade of tremendous, manual effort, John White et al. fully reconstructed 302 neurons and its thousands of synapses in the roundworm *Caenorhabditis elegans* which was stained with osmium, sliced into thin sections, and then imaged with a TEM (White et al. 1986) – the first connectome.

Important advances in the automation of volume electron microscopy (VEM) resulted from the development of block-face imaging techniques. In serial block-face scanning electron microscopy (SBEM), the brain sample is prepared as a single epoxy block, and the acquisition alternates between imaging and removal of the surface of the specimen block. The top layer of the block is thereby either cut with a diamond knife (DiK-SBEM) (Denk and Horstmann 2004; Leighton 1981), or ablated with a focused ion beam (FIB-SBEM) (Heymann et al. 2006; Knott et al. 2008). An alternative approach to block-face imaging is the automatic tape-collecting ultramicrotome (ATUM), where the sample block is cut into thin sections, collected on tape and mounted on wafers for the imaging with an SEM (Hayworth et al. 2006; Kasthuri et al. 2015).

SBEM approaches substantially reduce the complexity of data registration and artifacts that occur with serial sectioning, and led to the acquisition and analysis of increasingly large data sets in different species (Briggman et al. 2011; Kornfeld et al. 2017; Scheffer et al. 2020; Schmidt et al. 2017; Wanner et al. 2016). Disadvantages of SBEM techniques are its destructive nature, which demands almost perfect reliability, low imaging speed and limited sample size (Kornfeld and Denk 2018).

With the multi-beam scanning electron microscope (MSEM) (Eberle et al. 2015), traditional SEM has been extended to multiple electron beams in parallel, which increases acquisition rate dramatically by almost two orders of magnitude. While this system was successfully combined with serial sections using ATUM (Shapson-Coe et al. 2021), a combination with diamond knife microtomes is difficult, and focused ion beam (FIB) removal rates are too slow to be a viable with fast imaging (Kornfeld and Denk 2018).

One promising attempt that enables block-face imaging with multi-beam microscopes is based on a gas cluster ion beam (Hayworth et al. 2020), which smoothly ablates the sample surface by bombardment with ion clusters at high rates. This approach can be seen as a combination of serial sectioning with a diamond knife and FIB-SEM, which instead of operating on a single surface, alternates between imaging and milling of multiple thick sections (hundreds of nanometers up to a micrometer) that are collected on a silicon wafer.

Volume electron microscopy in connectomics is a tool to comprehensively reveal the fine structures of nerve cells and their connectivity. A study by Holler et al. 2021 extended it with functional light microscopy and electrophysiological recordings and verified a positive correlation between synaptic area and strength in mouse cortex. The wide improvements in speed and reliability of the image acquisition have led to insights into the presence of Hebbian-type plasticity (Bartol et al. 2015; Dorkenwald et al. 2021; Kornfeld et al. 2020; Motta et al. 2019) and the change of synaptic connectivity during development (Gour et al. 2021; Witvliet et al. 2021).

Groups at the Allen Institute and Harvard presented imaging pipelines that enable the acquisition of data sets in the petavoxel range (10^{15} three-dimensional pixels) using ATUM with TEM (Consortium et al. 2021; Yin et al. 2020) and MSEM (Shapson-Coe et al. 2021). With the whole mouse brain, the next milestone has already been set, a volume 10^{7} times larger than that required for *C. elegans* (Abbott et al. 2020).

1.2 Scanning electron microscope

Similar to light microscopy, lenses are a fundamental component for the imaging with electrons. In contrast to light, an electron beam can only be maintained in vacuum and therefore requires an alternative approach to solid materials that are otherwise used to alter the refraction index for focusing. Electrons that move inside a magnetic field experience a deflection orthogonal to it and their trajectory. This effect is described by the Lorentz force and realized in electron microscopes by short coils which produce axial symmetric, non-uniform magnetic fields that focus the electron beam (Egerton 2005, Chapter 2.3).

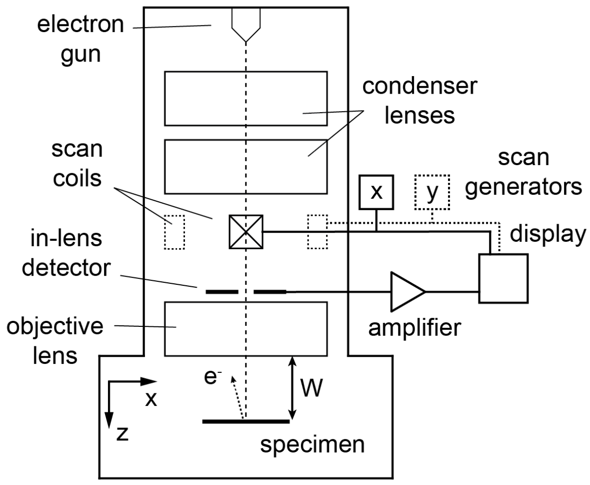


Figure 1.1: Schematic of a scanning electron microscope. W: working distance. Illustration inspired by Egerton 2005, p. 126.

The force that acts on an electron with charge $-e$, moving inside a magnetic field $\mathbf{B}$ (zero electric field component $\mathbf{E} = 0$) with velocity $\mathbf{v}$, is written as the cross product ($\times$) of the two vectors:

$$F = -e\,(\mathbf{v} \times \mathbf{B}) \tag{1.1}$$

The resulting force directly scales with the electron speed and preserves its magnitude, which allows focusing of a wide electron energy range and causes fewer aberrations compared to electrostatic lenses. An additional azimuthal component leads to spiraling of the electrons during the process of focusing, which results in a rotation of the image when the working distance is changed.

As the radial component of the magnetic field (perpendicular to the optical axis) is relevant to focus broad electron beams, a strong inhomogeneity of the magnetic field is necessary. This property is realized by short coils and strengthened by a ferromagnetic enclosing and so-called polepieces that concentrate the magnetic flux within a small volume at the optical axis. The strength of the magnetic field and thereby of the focusing is controlled by the direct current applied to the coil.

Electrons emitted from the electron gun in the microscope are demagnified with condenser lenses and finally focused with the objective lens onto the specimen (Fig. 1.1), which is usually done by the EM operator adjusting the working distance (W). Typically, a Schottky emitter is used as electron source which inherits long life times, low electron energy spread and a high brightness (Goldstein et al. 2003, pp. 34–35).

Two generators produce staircase waveforms at different frequencies to change the deflection of the beam periodically in the x-y plane through scan coils, thus scanning the specimen surface in a rectangular pattern. The beam-specimen interaction leads to backscattered and secondary electrons that can be measured with a detector. Binning the amplified detector signal with the scan waveforms finally results in the image of the scanned area.

Outer-shell electrons of specimen atoms that are released by the primary beam electrons through inelastic scattering are called secondary electrons (SE). The small remainder of energy transferred by the scattering results in an escape depth of only a few nanometers (Goldstein et al. 2003, Chapter 3.4) – the depth at which secondary electrons can still escape into the vacuum and potentially arrive at the detector. The scattering of the primary electrons intensifies with higher density and atomic number of the material in the interaction volume, which leads to a decrease in the electron range (Kanaya and Okayama 1972). As a result, areas in biological samples with more heavy metal stain lead to stronger signals and vice versa.

The focused electron beam impinging the surface of the specimen, also referred to as electron probe, can be described with four key properties, which are the diameter of the electron probe (also probe or spot size), the probe convergence angle (half the opening angle of the converging electron cone), the probe current and acceleration voltage of the beam. These parameters are adjusted by the SEM operator depending on the specimen, required image properties, and the type of the detected electrons.

Electron lenses suffer from defects that need to be taken into account to enable optimal imaging conditions (Goldstein et al. 2003, Chapter 2.3). With spherical aberration, electrons that are farther apart from the optical axis are bent to a point F_1 that is closer to the center of the objective lens, leading to an increased spot size in the image plane at location F, also called disk of confusion. The diameter of the disk of least confusion, which resides between the locations F_1 and F, depends on the maximum angle of the focused electrons α and a constant C_s being the coefficient of spherical aberration:

$$d_s = \frac{1}{2} C_s \alpha^3 \tag{1.2}$$

The effect of spherical aberration can thus strongly be reduced with an objective aperture that enforces small convergence angles. This however lowers the probe current and causes an increase in aperture diffraction:

$$d_d = \frac{0.61\lambda}{\alpha} \tag{1.3}$$

with λ being the wavelength of the electrons.

Similar to light microscopes, an electron beam with a non-zero energy spread will be focused at different locations, leading to chromatic aberration. The resulting disk of least confusion diameter is calculated by:

$$d_c = \alpha C_c \left(\frac{\Delta E_0}{E_0} \right) \tag{1.4}$$

with the electron energy E_0, energy spread ΔE_0 and the chromatic aberration coefficient C_c. Electron sources with low energy spread and large acceleration voltages are the means to counteract this aberration type.

Finding a set of reasonable beam parameters is therefore a trade-off between the different resolution-limiting effects. With fixed beam parameters, the location of the minimal aberration disk of the electron probe is found by adjusting the working distance of the microscope.

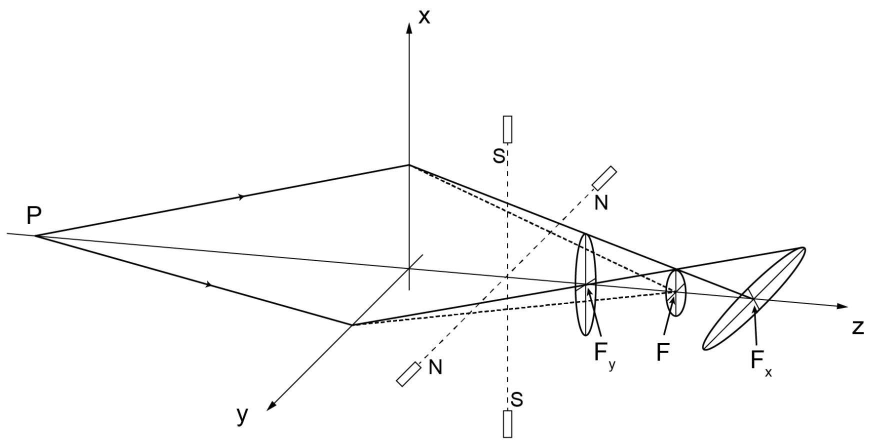


Figure 1.2: Effect of axial astigmatism on the electron beam (solid rays, originating from P) and its correction (dotted lines) with an electromagnetic stigmator based on a magnetic quadrupole. The beam forms an ellipse at F_x and F_y, and a circle at F. Illustration inspired by Egerton 2005, p. 52.

Any deviations from a cylindrically symmetric magnetic field in the lens will result in a focusing power difference depending on the plane of incidence of the electrons, also called axial astigmatism (Egerton 2005, Chapter 2.6). An astigmatic lens leads to stretching of the electron probe that at locations F_x and F_y can be approximated by two orthogonal ellipses, or line foci in the ideal case, one for the minimal and one for the maximal focusing power, respectively (Fig. 1.2). Although the stretching may be isotropic at location F, the probe size will be much larger than what would be optimally possible.

Axial astigmatism is caused, among other things, by machining inaccuracies of the pole-piece geometry but can be corrected by supplement magnetic fields, for example, provided by a magnetic quadrupol, which brings F_x and F_y together (Fig. 1.2). In practice, two quadrupols are used to adjust the beam with two control parameters (x- and y-stigmator).

The optimal probe size for imaging is finally found by adjusting both stigmators and the working distance of the microscope. This can be done manually with some training and experience, but long and continuous acquisitions in volume electron microscopy require alternatives.

1.3 Convolutional neural networks

With the increasing throughput of imaging techniques in volume electron microscopy, more and more data needs to be processed. One powerful tool for the processing of images are convolutional neural networks. Their origins can be traced to Hubel and Wiesel, who published a research paper in the 1960s about receptive fields of neurons in the visual cortex. They described two types of cells, one responsible for primitive feature detection and the other, with more complex input patterns, dedicated to direction-selective motion detection and being less sensitive to the spatial location of particular features (Hubel and Wiesel 1962). Inspired by this working principle, Fukushima and Miyake proposed the "neocognitron", an artificial neural network for pattern recognition and demonstrated its use on the identification of stimulus patterns of digits (Fukushima and Miyake 1982).

With the application of stochastic gradient descent and backpropagation (Rumelhart et al. 1986), a solution was found to efficiently change the network weights, which led to the development of convolutional neural networks (CNNs) and their successful application to digit recognition (LeCun et al. 1989). Key properties of a CNN are its sparse connections, parameter sharing and equivariance to input translation (Goodfellow et al. 2016, Chapter 9), which allows it to efficiently learn the detection of features. This is in strong contrast to fully connected feedfoward networks which form all-to-all connections.

The processing units (nodes) in a CNN are arranged as stack of layers, each representing pixel grids, where every node connects to a small and local subset of nodes (receptive field) in the previous layer. Every connection is parametrized by a weight and the receptive field of a node with its connection weights is represented by the convolution kernel. The number of kernels from one layer to another is configurable and, importantly, the weights of the kernels are shared across all nodes of a layer, which drastically reduces the number of parameters and consequently increases training speed. The linear output of a convolutional layer is then transformed by a non-linear activation function and a pooling operation, which reduces the resolution of the layer output by aggregating outputs within small patches.

In 2012, Alex Krizhevsky et al. presented a CNN that outperformed previous approaches in an image classification challenge by a large margin, a breakthrough that initiated the advent of deep learning in computer vision (Krizhevsky et al. 2012). Deep learning and its applications in the image domain, benefited further from a wide range of improvements, such as the use of rectified linear units (ReLU) (Glorot et al. 2011) as activation functions and batch normalization (Ioffe and Szegedy 2015), both significantly increasing training speed. Residual networks enabled backpropagation in very deep networks (He et al. 2016) and the U-Net represents an efficient approach to dense segmentation of volumetric images (Ronneberger et al. 2015).

1.4 Automated cell segmentation

Before EM image data sets can be processed and analyzed, the individual 2D images need to be consolidated by image registration. The data set essentially is a stack of consecutive two-dimensional images of the tissue sample, which in turn, due to a limited field of view (FOV) of the microscope, usually consist of multiple tiles. In a first step, these tiles are stitched in the x-y image plane, requiring a minimum amount of overlap between neighboring tiles. The stitched images are then aligned in z, orthogonal to the image plane, to correct shifts and artifacts that occurred during the image acquisition and finally stored as a coherent volume in a format that enables efficient access for processing and visualization.

With the availability of increasingly large data sets, a lot of effort has been put into developing software for efficient visualization and collaborative annotation (Helmstaedter et al. 2011; J. S. Kim et al. 2014; Maitin-Shepard et al. 2021; Schneider-Mizell et al. 2016; Sommer et al. 2011), with the goal to reduce the time requirement of the synaptic wiring reconstruction. Advances in compute hardware led to the beginnings of automated neuron reconstruction through algorithmic identification of pixel-level cell boundaries using artificial neural networks (Ciresan et al. 2012; Jain et al. 2007; Turaga et al. 2010).

The goal of neuron reconstruction is to assign all voxels in the volume either to a neuron segment entity or background, which can be seen as an instance segmentation of two classes (cell vs. background). These segments are represented by unique identifiers and are called supervoxels. The target of cell segmentation is to reconstruct the shape of neurons in EM image data, but it in general also includes non-neuronal cells such as glia.

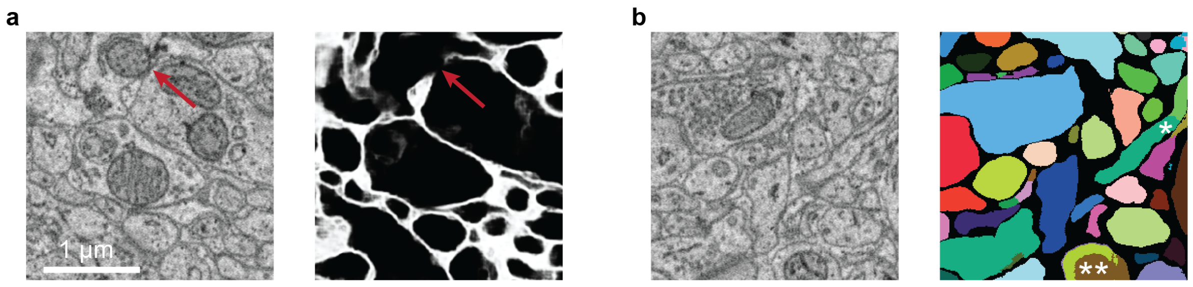


Figure 1.3: Cell boundaries and cell segmentation in image data acquired with a volume electron microscope (*area X zebra finch, small*, Appendix A). **a** An EM section and an example of the corresponding intracellular (black) and boundary regions (white) predicted by a convolutional neural network (Dorkenwald et al. 2017). The arrow indicates a missed membrane in the foreground prediction, potentially leading to a merge error in the neuron segmentation. **b** Neuron (over-)segmentation. Asterisks indicate example locations where a neuron is split into multiple fragments. Colors correspond to different cell fragments (supervoxels).

One way to approach cell segmentation is the aforementioned detection of boundaries between cells (Fig. 1.3a), which includes cell membranes and extracellular space. In an ideal case, where the boundary perfectly separates all cells, the intracellular volume could be algorithmically filled starting from any location inside the cell. The cell shape would then be represented by a single supervoxel. However, imperfections in the boundary (Fig. 1.3a), even if it is only a gap with the width of a single pixel, will result in merge errors, i.e. parts of two different cells will be covered by the same supervoxel.

To reduce the rate of merge errors, a so-called oversegmentation is generated to increase the reliability of supervoxels at the cost of splitting a cell into multiple, smaller fragments (Fig. 1.3b). This practice results from the fact that correcting a merge error post-hoc is very costly in contrast to solving a split error. Splitting a supervoxel requires the definition of a split surface and subsequent relabeling of all involved voxels, while merging can be done by assigning two fragments the same identifier, i.e. the same cell.

In a next step, the supervoxels are agglomerated to form proposal reconstructions of entire cells. For this purpose, supervoxels can be represented as vertices of a graph, and supervoxels likely belonging to the same neuron (reconstruction) can be linked through edges, leading to a supervoxel graph. If edges in that graph contain weights based on the underlying agglomeration method, subsequent thresholding is applied to yield the final segmentation (supervoxel agglomeration), where each connected component represents one cell.

Flood-filling networks (Januszewski et al. 2018), the current state of the art to neuron segmentation, are based on a convolutional neural network architecture with a recurrent pathway, which learns to iteratively fill the intracellular region of individual neurons. The prediction starts with an empty mask at sampled start (seed) locations and terminates if the filled cell is surrounded by a closed background prediction. Cells are "oversampled" with multiple seed locations leading to an initial oversegmentation that is subsequently refined by a consensus strategy, which for example includes the application of multiple models with different input data resolutions (voxel sizes). Januszewski et al. 2018 reported striking results with an average path length of 1.1 mm before encountering an error. Considering data sets with volumes in the range of hundreds of microns side length, the resulting reconstructions thus closely match the neuron shape contained in the volume.

Aligned EM image data together with the neuron segmentation is the starting point for connectomic analysis. Fig. 1.4 shows the shape reconstruction of a neuron consisting of three anatomically distinct compartments, the axon, soma, and dendrites. The neuron shape can be extended by identifying neurobiologically relevant ultrastructure captured in the EM images, such as mitochondria, synaptic vesicles and synaptic junctions (Fig. 1.4 left). The detection of individual synapses and involved cell partners as well as methods for the analysis of cellular morphology allow a comprehensive reconstruction of neurons and their synaptic wiring.

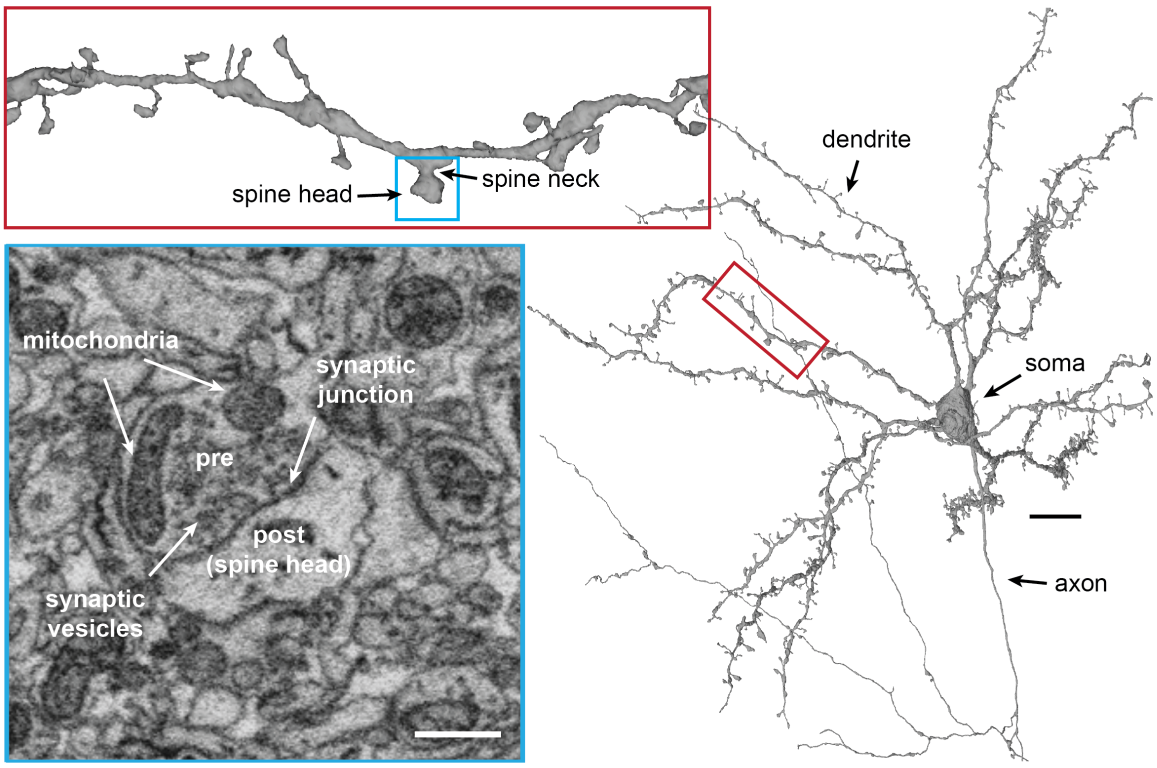


Figure 1.4: **Left**: EM section showing the cellular ultrastructure of a presynaptic axon (pre) forming a synapse with a spine head (post, postsynaptic) of the cell visualized on the right. **Right**: Shape reconstruction of a medium spiny neuron generated by flood-filling networks in the zebra finch brain region area X (*zebra finch area X, large*; Appendix A). Scale bars are 500 nm in the EM section and 10 µm in the rendering.

1.5 Objectives and contributions

The goal of this thesis was to advance volume electron microscopy image acquisition and to automate the analysis of the generated data sets further.

Image acquisition

Since GCIB is one of the most promising technologies for volume electron microscopy data acqusition, it became one of the objectives of my thesis to set up an imaging pipeline for thick, heavy metal stained sections with gas cluster ion beam milling. Chapter 2 presents a prototypical procedure for the collection of thick sections cut with an ultramicrotome and outlines the imaging procedure for hundreds of sections.

An essential part of highly-automated image acquisition is to maintain good image quality in terms of focus during data acquisition with a scanning electron microscope. The automation of focusing and stigmation was realized with a deep learning based approach using a convolutional neural network.

Automated data analysis

The reconstruction of neuron shapes is not sufficient for a comprehensive circuit reconstruction and increasingly large data sets become available. Hence, the third objective was to increase the degree of automation of the steps that follow cell and ultrastructure segmentation and provide a solution to circuit reconstruction. Chapter 4 describes methods for the detection of anatomical compartments in neurons, that also allow the identification of pre- and postsynaptic partners, supervised cell type classification and unsupervised clustering of neurons. Chapter 5 presents the *SyConn2* connectome analysis framework for automatic circuit reconstruction based on volumetric electron microscopy images and corresponding cell segmentation.

Chapter 2

Wafer-based image data acquisition with GCIB

2.1 Introduction

Serial-section electron microscopy is based on thin sections ($\sim 50\,\text{nm}$) that are cut by the thousands with an ultramicrotome and collected on a carrier material, for example tape (Schalek et al. 2011) or silicon wafers (Horstmann et al. 2012). However, folds, cracks and the loss of entire sections make it difficult to scale this technique up to large volumes without defects (Macrina et al. 2021; Shapson-Coe et al. 2021; Zheng et al. 2018).

MSEMs (Eberle et al. 2015) have recently been combined with an automatic tape-collecting ultramicrotome (Shapson-Coe et al. 2021), but are incompatible with serial block-face approaches (Hayworth et al. 2020; Kornfeld and Denk 2018). Hayworth et al. 2020 introduced gas cluster ion beam (GCIB) milling for volume electron microscopy that allows, unlike FIB milling, the ablation of large areas and demonstrated the feasibility of acquiring image stacks. They used up to 1 µm thick sections, which is one order of magnitude larger than what is usually used with serial-section electron microscopy. For once, this results in an overall lower section count and second, in a small image stack (hereinafter called sub stack) instead of a single image per section.

All sub stacks need to be sorted in case they were not tracked during collection to retrieve the final EM volume. The sorting is less difficult as with thin sections, because the correlation between two images declines with increasing z-distance (orthogonal to the image x-y coordinates) and the total section count for the same volume is lower. As a consequence, the correlation between first and last slice of two adjacent sub stacks will be much higher relative to other sub stacks (GCIB milling) than the correlation between two images of ultra-thin sections that are only $\sim 50\,\text{nm}$ apart[1].

When combining GCIB milling with (multi-beam) scanning electron microscopes, it is favorable if the milling time takes up only a small portion of the imaging time – otherwise

[1]Hayworth et al. 2020 estimated the loss between two consecutive thick sections that were cut with an ultramicrotome to be $\sim 30\,\text{nm}$

the effective imaging speed will be slowed down accordingly. To this end and assuming a constant mill current, it is important to ensure a high packing density of sections, which in addition to the aforementioned issues, is not directly given with tape-collected sections.

Templier 2019 demonstrated a reliable wafer-based collection method of hundreds of thin sections in which the sample was re-embedded into a resin block that contained superparamagnetic nanoparticles. An actuated magnet allowed to move the floating sections in the knife boat after cutting with an ultramicrotome. The sections were confined above a silicon wafer and adhered to the wafer after lowering the water level. For the collection of 507 sections he observed tears in 9 sections and no section loss, which becomes particularly relevant with thick sections. Although densely packed, the supplement epoxy block reduces packing efficiency and can further exacerbate dulling of the knife, especially with large samples, a common problem with serial sectioning.

This chapter presents an alternative, prototypical wafer-based collection procedure for thick sections and milling experiments with an argon-carbon dioxide gas mixture instead of pure argon used by Hayworth et al. 2020.

2.2 Materials and Methods

Experimental setup

All experiments used a Zeiss field emission SEM (Merlin) for the generation of EM images and a GCIB 10S ion gun from Ionoptika to mill the specimen surface (Fig. 2.1). A high-pressure gas connection supplies the horizontally oriented gun with an argon-carbon dioxide gas mixture (18% CO_2 and 82% Ar; Corgon-18 from Linde). The gun is connected to the vacuum chamber of the EM through a manual valve and two additional turbo pumps (Pfeiffer Vacuum TM 700) with dampers to provide a stable vacuum without introducing vibrations into the system.

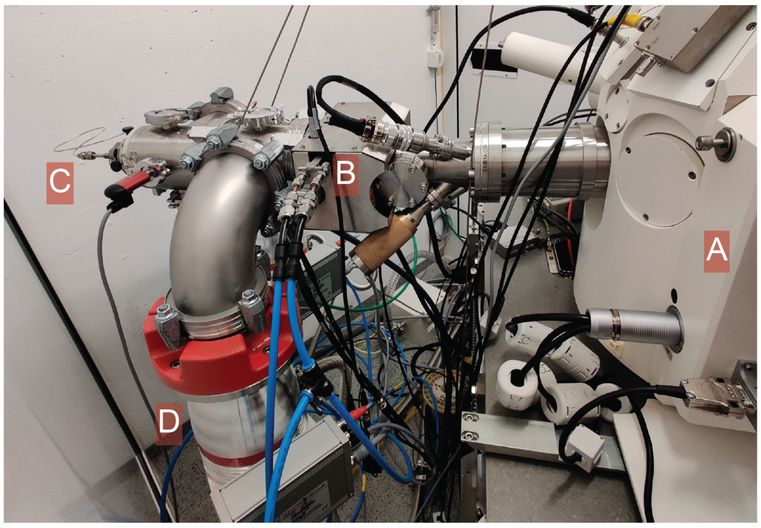


Figure 2.1: Experimental setup of the single beam field emission SEM (A) together with an ion gun (B). The ion gun is connected with a high-pressure gas supply (C), two turbo pumps (D) and the imaging chamber of the EM.

For the acquisition process the open source software SBEMimage[2] (Titze et al. 2018) was extended to support ion gun milling. Between two consecutive imaging iterations, the stage together with the specimen is moved to the center of the ion beam and rotated to an adjustable incident angle. To mill the sample surface from multiple directions for uniform material removal, the stage is rotated about the vertical stage axis in azimuth direction for a specified milling duration. The rotation was divided into three different azimuths separated by 120 degrees with a 10 s delay before rotating to the next azimuth. The SEM focus was maintained using the open source Python implementation of MAPFoSt (Binding

[2]https://github.com/SBEMimage/SBEMimage

et al. 2013) by R. Saxena[3] with five correction iterations.

A secondary electron in-lens detector provided the imaging signal using a 1.5 kV acceleration voltage, a probe current of 1.8 nA and 800 ns pixel dwell time in all experiments. The ion gun current was 38 nA, measured with the built-in current detector of the used Zeiss SEM. The ion beam glancing angle was set to 30 degrees and ion clusters were tuned to a cluster size of 2,000 ions with an acceleration voltage of 10 kV and a slight defocus of the beam for smooth milling.

The scan pattern of the ion beam was controlled with a waveform generator (Agilent 33522A) that generated two superimposed sawtooth voltages, where the amplitude of each signal controlled the spatial extent of the beam in one of two orthogonal directions (peak-to-peak voltage range: $0\,V_{pp}$ to $20\,V_{pp}$). A frequency of 100 Hz and 1 Hz was used for X and Y, respectively. The irradiated area was calculated from the applied scan voltages using the LineScan tool provided by Ionoptika together with a calibration target (a washer with known diameter).

Section collection and preparation

For the extraction of the tissue sample, a male zebra finch (days post hatch: 124) was anesthetized with sodium pentobarbital, distributed under the name Narcoren (Böhringer Ingelheim), and perfused with an extracellular space preserving solution containing 0.07 M cacodylate (Serva) buffer, 0.14 M sucrose (Sigma-Aldrich), 0.002 M $CaCl_2$ (Sigma-Aldrich) and a subsequent fix solution, extending the previous solution with 2% paraformaldehyde and 2% glutaraldehyde (Serva). After carefully extracting the brain it was post-fixed at 4° overnight and then put in 0.15 M cacodylate buffer. The brain was cut with a vibratome (Thermo Scientific HM 650V) in 250 µm thick sections and a biospy punch (Leica Biosystems) was used to extract a $500 \times 500 \times 250\,\mu m^3$ basal ganglia sample. The sample was stained as described in Hua et al. 2015 and embedded with Spurr's (EMS) by A. Rother.

The sections for the mill rate estimation were cut from the sample with an ultramicrotome (Leica EM UC7) and a diamond knife (DIATOME ultra jumbo), collected manually with a Perfect Loop (Diatome) and placed on gold coated (Leica EM MED020 with gold rods) silicon wafers. If not stated otherwise, all section had a thickness of 200 nm.

For the automated section collection, a floating polyester film (floater, RS PRO Mylar A with 0.25 mm thickness) was used to confine the area for the floating sections in the knife boat of the ultramicrotome. The floater was cut to the required shape with a CO_2 laser (ILS12.75 from Universal Laser Systems). After the sections were cut and confined above the wafer, the water level was slowly reduced using a pipette. The wafer was then placed on a heating plate to dry and support flattening of the sections at approximately 60 °C. Silicon wafers were treated in advance with a plasma cleaner (PDC-002 from Harrick Plasma) for about one minute to promote wetting.

[3]https://pypi.org/project/mapfost/

The diameter of the wafers (p-type, J11002 from SIEGERT WAFER) was 25.4 mm for all experiments.

Collected thick sections were further irradiated with a strong electron source (EH-50 STAIB instruments) to increase conductivity, which is necessary to enable artefact-free imaging (Hayworth et al. 2020). The electron energy density was monitored during the irradiation with a current preamplifier (SR 570 from Stanford Research Systems) and a digital multimeter (Keysight 34465A) using a custom Python script. The current measured by the preamplifier was integrated (Simpson's rule) over time (1 s intervals), yielding q_{tot}, to calculate the deposited energy volume density given the irradiated area A and thickness d of the section:

$$\rho_E = \frac{U_{beam} \cdot q_{tot}}{A \cdot d} \tag{2.1}$$

For 200 nm thick sections the beam acceleration voltage was set to 3 kV and to 5 kV for 500 nm.

Computational section processing

The acquired image stacks for the mill rate experiments were aligned with the SIFT (Lowe 2004) plugin from Fiji (Schindelin et al. 2012). As the sample surface gets milled away successively, the wafer becomes visible. Due to inhomogeneities in milling, the breakthrough occurs after a different number of iterations (z) for different locations (x-y). To identify the first wafer occurrence in z for every pixel in x-y the following procedure was applied: the distinct wafer signal (low pixel intensities) was segmented by thresholding the gray-value image pixels (0..255) with an appropriate threshold (< 10). The z-index for the last sample encounter was found by arg min of the inverted (top-bottom) wafer segmentation column; in the inverted column of the wafer segmentation the first 0 value occurs at the originally last z-slice of the sample.

The height indices were extracted from a square of 1000×1000 pixels for every sample, slightly smoothed using a Gaussian kernel with a standard deviation of 1. The average and the error of the number of milling cycles was then calculated by taking the mean and the standard deviation of all indices. The standard deviation of the resulting mill rates was found by propagation of uncertainty. The identification of the last sample slice was done manually by inspecting the aligned tiles and finding the z-slice with minimal debris and no substantial change thereafter.

For the automatic section detection on the wafer a CNN with a 2D U-Net (Ronneberger et al. 2015) architecture was employed using 5 blocks, 64 initial filters, ReLU activation (Glorot et al. 2011) and batch normalization (Ioffe and Szegedy 2015) to discriminate sample tissue from background, which included the wafer and the epoxy embedding of the sample.

With the help of a custom Python script based on the image annotation tool Napari (Sofroniew et al. 2021), ground truth for a two-class segmentation was generated (0: background, 1: sample tissue) in a sparse fashion. An auxiliary ignore label was used for unlabeled regions (2: ignore), which did not contribute to the loss calculation during training. Labeled segments of a single class were outlined by drawing polygons on the wafer overview mosaic (Fig. 2.2). Background labels were added automatically around the tissue class by dilating (binary morphological operator) label-1 regions 10 times into ignore regions and converting the dilated area to background (Fig. 2.2 center). The wafer overview was downsampled to a pixel size of 4 µm for training and inference.

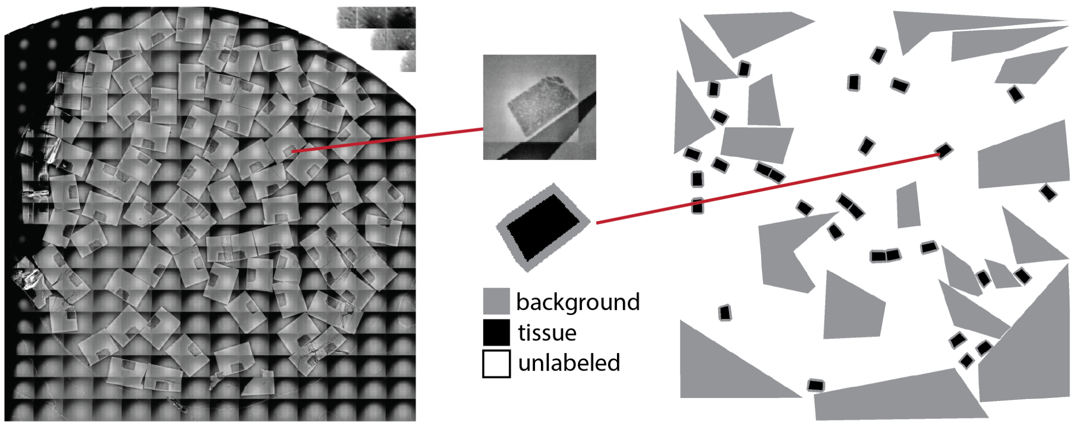


Figure 2.2: Sparse ground truth annotation for section detection on silicon wafers. **Left:** A Mosaic overview of size 13.92 × 14.01 mm^2 (14 × 19; 32 pixels overlap) acquired with SBEMimage, which contains 87 collected thick sections of a 500 × 250 µm^2 sample on a wafer. Individual image tiles were taken with 1024 × 768 pixels, 1 µm pixel size and 200 ns pixel dwell time. **Center:** Background labels around a labeled thick section generated by binary dilation. **Right:** Partially annotated target labels for automatic section detection.

The model inputs were patches with 256 × 256 pixels, grayscale intensities between 0 and 255 and were centered and rescaled to -0.5 and 0.5. The following augmentations modified the input images I during training:

- Shift, scale and rotation applied with probability $p = 0.95$. Relative shift values were drawn from $\mathcal{U}(-0.0625, 0.0625)$, scale from $\mathcal{U}(-0.2, 0.2)$ and rotation angles from $\mathcal{U}(-180, 180)$. The resulting affine matrix was applied with cubic interpolation and reflecting boundary mode.

- Random flips applied with probability $p = 0.5$, independently on x- and y-axis.

- Elastic transformation based on (Simard et al. 2003) with probability $p = 0.5$, smoothing parameter $\sigma = 2$ for the displacement field and a scaling parameter to control the displacement strength $\alpha = 5$.

- Additive Gaussian noise applied with $p = 0.75$: $I^* = I + X$ with X being a 2D noise map with the same size as I and i.i.d. $X_{i,j} \sim \mathcal{N}(0.0, 0.1^2)$.

- Random gamma adjustment: $I^* = I^\gamma$ with $\gamma \sim \mathcal{N}(1.0,\, 0.25^2)$ and $p = 0.75$. Pixel intensities are internally rescaled between 0 and 1.
- Random brightness B and contrast C adaption: $I^* = C(I - I_{mean}) + I_{mean} + B$ with $C \sim \mathcal{N}(1.0,\, 0.25^2)$ and $B \sim \mathcal{N}(0.0,\, 0.25^2)$.

The deep learning framework elektronn3[4], based on PyTorch (Paszke et al. 2019), was used for model training, augmentations and tiled inference. The shift-scale-rotate augmentation was imported from the open-source package Albumentations (Buslaev et al. 2020).

Training was performed with AdamW (Loshchilov and Hutter 2019) optimizer (initial learning rate 0.001, weight decay $0.5 \cdot 10^{-4}$), learning rate scheduler (scheduler step size of 1000, decay 0.9), cross-entropy loss (class weights: 1, 2) and was stopped after approximately $1.8 \cdot 10^5$ iterations (18h training time on one Nvidia Quadro RTX 5000). Predictions on ground truth pixels with label 2 were ignored during loss calculation.

The tiled prediction (200×200 input patches with additional 30×31 pixels overlap) of the mosaic overview was processed by applying one iteration of binary closing, followed by a Canny edge filter, contour detection and a minimum-rectangle extraction from the opencv2 Python package (https://github.com/opencv/opencv-python). In addition, individual foreground instances in the binary-closed prediction were identified via connected components (`ndimage.label` from the scipy (Virtanen et al. 2020) package). Rectangles with a pixel area larger than 200 were assigned to a foreground instance by using their center coordinate.

If a region of interest (ROI) was larger than the field of view of an undistorted SEM scan, it was divided into multiple tiles, further called grid, where each tile is represented by one scan. All resulting rectangle instances were loaded into SBEMimage and used to instantiate such acquisition grids, which also allows subsequent adjustment and placement of missing or removal of wrongly identified grids.

For the minimization of the stage movements, a heuristic optimization implemented in the python-tsp Python package (https://github.com/fillipe-gsm/python-tsp, with $\alpha = 0.9$ and 2-opt perturbation scheme) and based on simulated annealing (Kirkpatrick et al. 1983) was run on the pairwise distance matrix between all grids. Distances back to the start grid were set to zero.

The computational flattening in Fig. 2.7 was performed using the pixel-wise height index as described above and linear 1d-interpolation to the full extent via the function `interpolate.interp1d` from the scipy Python package (Virtanen et al. 2020).

[4]https://github.com/ELEKTRONN/elektronn3

2.3 Results

2.3.1 GCIB mill rate estimation

While tape-collection approaches allow quality control with potential re-imaging in case of acquisition artefacts, the z-resolution is limited by the ultramicrotome to at least 30 nm. Using thick sections instead, this limit can be reduced to less than 10 nm with gas cluster ion beam milling (Hayworth et al. 2020), which allows smooth surface ablation for parallel[5] block-face SEM.

To test this approach, a field emission SEM setup was extended with an io gun from Ionoptika. The gun was connected to a high-pressure supply for an argon and carbon dioxide gas mixture. Upon release into the evacuated extraction chamber the gas undergoes adiabatic expansion and forms clusters, which are ionized, accelerated and then tuned to a specific size with a Wien filter. Two beam deflectors enabled scanning and the adjustment of regions of interest on the specimen.

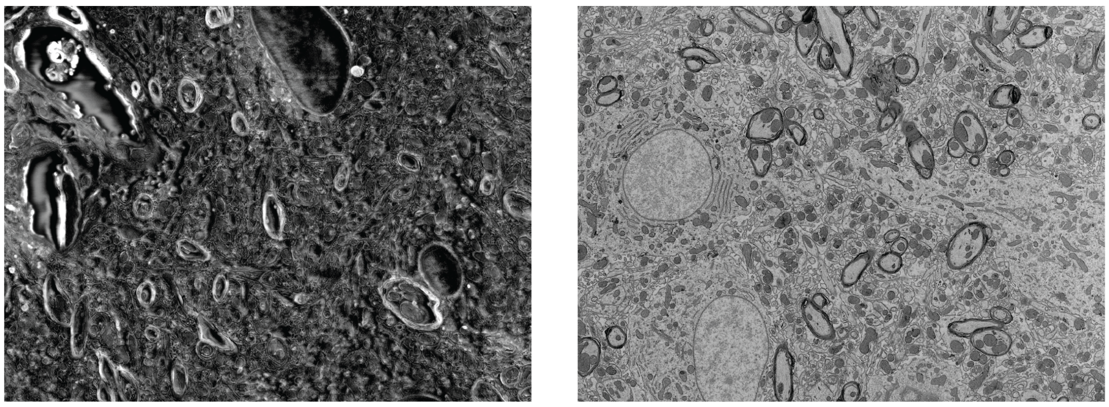


Figure 2.3: Effect of electron irradiation on charging artefacts during SEM imaging of two sections with 200 nm thickness. **Left:** Section without prior electron irradiation. **Right:** Section that was irradiated with a total dose of $0.5\times10^{27}\,\mathrm{eV\,cm^{-3}}$. Note that the images contain different sample areas.

Increasing the thickness of sections reduces electrical conductivity to a level that leads to severe artefacts during imaging, which can be counteracted by electron irradiation at the cost of a reduced mill rate (Hayworth et al. 2020; Fig. 2.3). The dependency of the mill rate on the amount of electron irradiation can be measured by milling specimens of known thickness at different electron irradiation levels. The experiment was conducted in analogy to the proof-of-concept study by Hayworth et al. 2020, but using a cheaper argon-carbon dioxide gas mixture (Corgon-18, 82% argon) instead of pure argon.

[5]In the sense that multiple thick sections are processed within a single milling iteration.

Table 2.1: Mill rate measurements for different electron irradiation levels. All samples had a thickness of 200 nm. FoV: edge length of the field of view of the ion gun scan, $\dot{V}$: mill rate, $\dot{V}/I_{beam}$: mill rate per applied beam current.

dose $\left[10^{27}\frac{\mathrm{eV}}{\mathrm{cm}^3}\right]$	T_{mill} [s]	FoV [µm]	$\dot{V}\left[\frac{\mathrm{µm}^3}{\mathrm{s}}\right]$	$\dot{V}/I_{beam}\left[10^{-8}\frac{\mathrm{m}^3}{\mathrm{C}}\right]$
1	31680	9462	565	1.487
0.5	16800	7885	740	1.948
0.25	10080	7885	1233	3.246
0	8640	9462	2072	5.453

For this experiment, thick sections were cut from a biological sample with an ultramicrotome, collected on silicon wafers and treated with a varying duration of electron irradiation. The acquisition was performed with the GCIB-SEM setup and automated by extending the open source software SBEMimage (Titze et al. 2018) with the necessary functionality to control the ion gun. For each sample, the final tissue slice was identified to calculate the average mill rate $\dot{V}$ from the total mill duration T_{mill} and the scanned sample area (Table 2.1).

The measured values for Corgon-18 are similar to those reported for pure argon in Hayworth et al. 2020. A mill rate of $\dot{V} = 740\,\mathrm{eV\,cm^{-3}}$ translates to 740 MHz assuming perfect area usage and isotropic 10 nm voxels, which is in the same order of the scan speed of an MSEM. This underlines the importance of either dense packing of sections on the wafer or a high effective milling speed to keep both rates in appropriate balance.

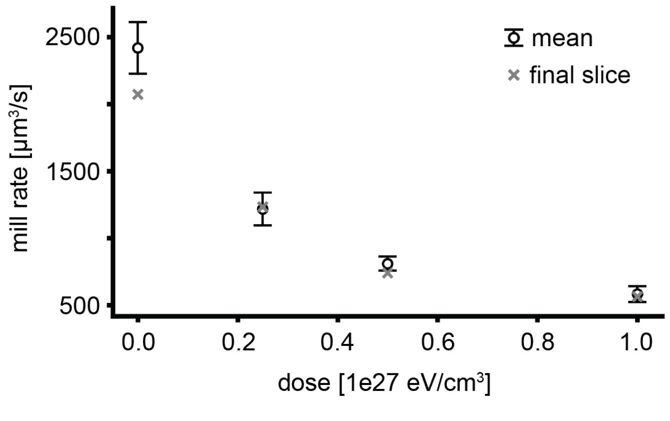


Figure 2.4: Average GCIB mill rate dependent on the electron irradiation for complete removal of 200 nm thick tissue sections obtained by finding the final tissue slice manually and as mean and standard deviation on pixel level (Materials and Methods). Images did not show charging artefacts with an irradiation dose of $10^{27}\mathrm{eV\,cm^{-3}}$ or more.

In addition, a segmentation of the surfacing wafer was performed in the acquired sub stacks to identify the number of mill iterations on pixel level. Except for the unirradiated sample, the pixel-mean mill rate was close to the mill rate obtained from the total milling time, determined by manual identification of the final slice (Fig. 2.4). Similar to Hayworth et al. 2020, simple thresholding was only possible with (gold-)coated wafers.

2.3.2 Volume acquisition scheme and post-processing

To automatically collect thick sections on a silicon wafer, a prototype collection method was designed which uses a floating polyester film in the microtome knife boat to confine the movement of cut sections to an area above the immersed wafer (Fig. 2.5). In contrast to Templier 2019, which requires the extension of the sample by a resin block filled with superparamagnetic nanoparticles, the presented approach does not require the modification of the sample block. This is relevant to GCIB-SEM as the milling speed depends on the treated area. Consequently, for efficient milling, the packing density of the target tissue on the wafer should be sufficiently high and minimize the non-tissue area.

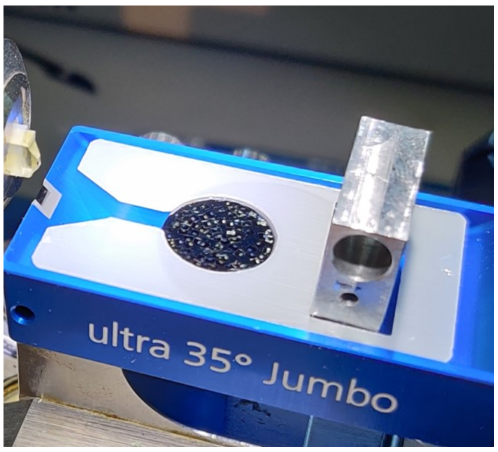


Figure 2.5: Section collection on a silicon wafer using a polyester film that confines the cut sections to an area above the wafer. Diamond knife boat with the sample block on the left.

Using this procedure, 87 thick sections were collected to develop methods for the next steps in the acquisition process. Unfortunately, several sections were lost due to cracking and folding when manually lowering the water level during the collection, illustrating the difficulty of automated section collection. This particularly affected peripheral sections, which needs to be addressed by future optimizations.

To structure the acquired image data and reduce the amount of imaged background, it is necessary to identify all regions of interest, i.e. the area containing tissue in every section, which becomes laborious and time-consuming when thousands of sections are involved. In order to semi-automate this task, regions of interest and background were sparsely annotated in a mosaic overview of the wafer (approx. 30 min total manual annotation time) and served as ground truth for segmenting the entire overview with a U-Net (Ronneberger et al. 2015) (Fig. 2.6). It turned out that adding a background boundary around the foreground masks was crucial to accurately segment every section (Fig. 2.2).

The foreground segmentation was further used to identify a bounding box for every section, which were in turn made available in the graphical user interface (GUI) of SBEMimage as acquisition grids. A grid tiles one region of interest into multiple scans and enables manual adjustments in the user interface before starting the acquisition. To reduce overhead in stage movements, the ordering of the grid traversal can be optimized by minimizing the total path length (Fig. 2.6 right). This might also have a positive effect on associated mechanical wear and reduces the settling time of the stage if neighboring sections are close (so that the stage does not reach maximal speed).

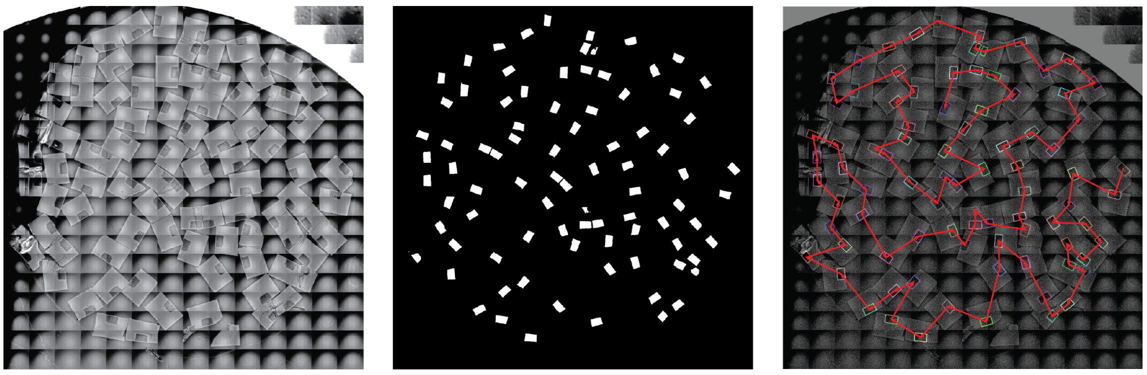


Figure 2.6: Preparations for the acquisition of thick sections collected on a silicon wafer. **Left:** Mosaic overview from Fig. 2.2. **Center:** Predicted sample regions that contain stained tissue. **Right:** Example section traversal during acquisition by greedy minimization of the traversed path length.

After the alignment of the acquired sub stacks, one for each each thick section, a computational flattening is necessary due to small, local inhomogeneities of the mill rate (Fig. 2.4, Fig. 2.7; Hayworth et al. 2020). The slices close to the wafer showed some imaging artefacts (see increased brightness in Fig. 2.7), presumably caused by hydrocarbon contamination due to accumulation of ablated material. This could be addressed by additional mild plasma cleaning, for example using the GV10x downstream asher (ibs) between the milling and imaging cycle. The partial stacks must be sorted into the correct z-order, e.g. by pairwise cross-correlation of the top and bottom slices of all partial stacks, followed by a final alignment, which is subject to future efforts.

An acquisition of 500 sections with 500 nm thickness and 0.25×0.25mm tissue area (in total 7.8 teravoxels, $0.015\,625\,\mathrm{mm}^3$) would require about 30 d (milling: 4 d, imaging: 26 d; irradiation: 4 d)[6] on a single-beam SEM (5 MHz scan rate, 200 ns pixel dwell time resp.), assuming a mean mill rate of $856\,\mathrm{s\,nm}^{-1}$ for an area with 22 mm side length, extrapolated from the value for an irradiation of $0.5 \cdot 10^{27}\,\frac{\mathrm{eV}}{\mathrm{cm}^3}$, a scan tile size of 6000×6000 pixels and pixel size of $10 \times 10 \times 20\,\mathrm{nm}^3$.

[6]For the calculation, 5×5 tiles per section are considered, leading to an effective imaging rate of about 3 MHz and a packing density of about 10% with a mean wafer area per slice of 0.63 mm.

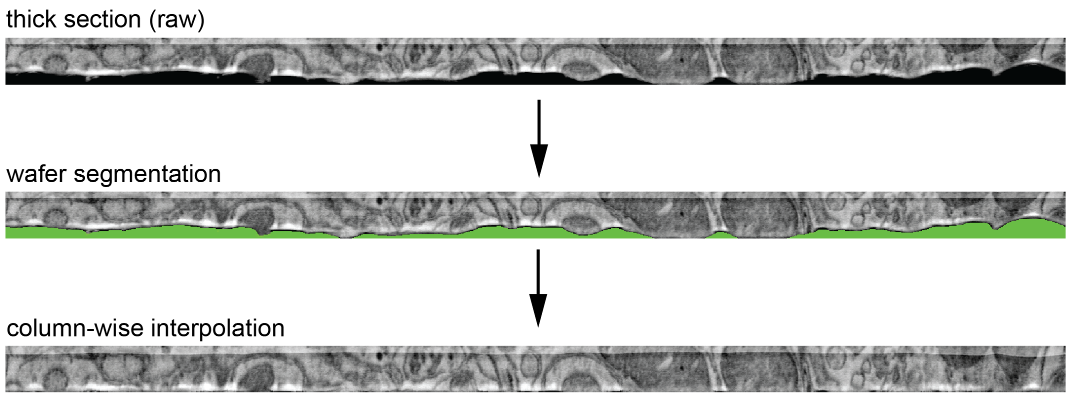


Figure 2.7: Computational flattening of a 500 nm thick section as proposed by Hayworth et al. 2020: The wafer signal in the initial image stack (top, black) is segmented using thresholding (middle, green) and used to linearly interpolate the sample signal to the full z-extent, independently for every z-column (bottom).

2.4 Discussion

Surface milling of stained and epoxy embedded neural tissue with a gas cluster ion beam for volume electron microscopy connectomics has recently been demonstrated (Hayworth et al. 2020) and represents a promising technique compatible with high-throughput multi-beam scanning electron microscopes. Here, a single-beam field emission SEM was extended with GCIB milling and was successfully applied to image ultramicrotome-cut thick sections. The mill rate with an argon-carbon dioxide gas mixture showed an electron irradiation dependency similar to what was reported in Hayworth et al. 2020 for pure argon. The GCIB acquisition routine was implemented in the open-source EM imaging tool SBEMimage, which might proof useful for future studies.

With further optimization, the presented approach for section collection could evolve into a simple yet effective procedure for future volume electron microscopy data sets that does not require modification of the sample epoxy block. Collecting sections for example with in-bath heating would allow a slow and controlled lowering of the water level until the sections adhere to the wafer. Thick sections in, contrast to thin ones, do not tolerate even a single loss, otherwise it would be impossible to unambiguously trace all neurons in the worst case. The collection of 2000 sections, e.g. for a 1 mm thick sample, with a success rate of $p = 0.1$ and without a single lost section, requires a loss rate of about 0.001 per cut, which makes highly reliable collection critical for wafer-based GCIB-SEM.

Chapter 3

Automated focus and stigmation correction

Key results of this chapter were submitted as patent EP21212051.3, which is currently pending (Schubert and Kornfeld 2021, December 2).

3.1 Introduction

In SBEM the sample undergoes no or only very small vertical displacements (in the order of tens of nanometers) during milling (focused ion beam) or cutting (diamond knife). Any resulting focus drift can either be corrected via iterative optimization (Xu et al. 2017), a heuristic autofocus (Briggman et al. 2011) or even manually in regular intervals. In contrast, data acquisition with GCIB milling requires stage movements in the centimeter range between two consecutive imaging cycles, which result in displacements of up to several micrometers due to imprecisions of the microscope stage motor (Hayworth et al. 2020). Correction of the imaging parameters is therefore more challenging.

In (Hayworth et al. 2020), the authors presented a simple ad-hoc solution based on a grid search around the last known, good focus parameters in order to correct the working distance and stigmator parameters before the acquisition of each image. This procedure worked well for the purpose of showing the general usability of GCIB in VEM with the acquisition of small volumes (on the order of 10 µm edge length), but was not optimized for speed. Meanwhile, for large volumes, the autofocusing and -stigmation procedure must be precise and very robust, and, at the same time, the processing overhead must not significantly affect the remaining acquisition.

With wafer-based acquisition, thick sections of tissue sample up to hundreds or thousands are collected on a silicon wafer. For each ROI, which usually is a grid of overlapping tiles for every thick section, beam parameters for a fine electron probe must be verified after every milling cycle. In combination with the destructive nature of this acquisition method, a reliable and fast autofocus procedure is critical.

In order to keep the additional electron dose (electrons per area) by the auto-focus

(AF) small, the imaging time of a given beam current on the same sample area needs to be minimized, which is necessary to prevent imaging artifacts through irradiation damage (Egerton et al. 2004) and local changes in the milling rate (Fig. 2.4, Hayworth et al. 2020). Minimization can be achieved by altering two factors: Firstly, by increasing the scan speed, which reduces the pixel dwell time and thereby also the image signal-to-noise ratio (SNR), and, secondly, by reducing the number of acquired test images for the correction estimation. The latter was optimized by a Bayesian approach called Maximum-*A-Posteriori* Focusing and Stigmation (Binding et al. 2013) (MAPFoSt), which requires the acquisition of only two test images.

Approaches that explicitly model the electron probe formation and aberrations (Binding et al. 2013; Erasmus and Smith 1982; Paxman et al. 1992) by taking test images or by using classical image sharpness scores (Batten 2000; Rudnaya 2011; Xu et al. 2017) generalize very well in theory and can be applied to different microscopes and specimens with very little or no modifications to their working principles. In practice, they still require careful, manual parameter tuning.

A recent study proposed an approach with two artificial neural networks to infer the quality of SEM images with a subsequent correction estimate for the working distance based on an updating state vector and a database of tens of thousands of manually labeled images (W. Lee et al. 2021). The excellent performance of CNNs in general image processing tasks and the success of deep learning models for focus correction in light microscopy (C. Li et al. 2021; S. J. Yang et al. 2018) provided the incentive to develop a deep learning based method, dubbed “DeepFocus”, for both focus and stigmation correction in SEMs that is fast, accurate and requires comparably little training data.

3.2 Materials and Methods

Electron microscopes and samples

All experiments were conducted on two Zeiss single beam SEMs: One was a Zeiss Merlin with 1.5 kV acceleration voltage, 1.5 nA beam current, in-lense secondary electron detector and a working distance of 4.5 mm (setup A). The recalibration experiments on setup B used a Zeiss UltraPlus SEM with, if not stated otherwise, 1.2 kV, 60 µm aperture, in-lense secondary electron detector, 6 mm working distance and a by 90° rotated scan. All EM images were acquired with 10 nm pixel size.

All experiments that involved a biological sample were carried out on 250 nm thick, electron irradiated (see Chapter 2, Hayworth et al. 2020) sections collected on a silicon wafer (sample and procedure as in Chapter 2). The non-biological samples were tin on carbon specimens from agar scientific (S1937) on both setups A and B.

The stigmator values were used as reported by the microscope software (SmartSEM Version 6.06) without additional adjustment or calibration.

Ground truth generation

Training and validation samples were generated by acquiring a pair of perturbed ($\pm\sigma_{wd} = \pm 5$ µm working distance) images relative to a known aberration, which introduced defocus by changing working distance δ_{wd} and both stigmators $\delta_{stigx,stigy}$ (Fig. 3.1). At a given location, the focus baseline was adjusted manually and perturbed image pairs were acquired for 10 introduced aberration vectors, that were drawn uniformly within a given value range $\delta_i \sim \mathcal{U}(a_i, b_i)$ (with i for working distance, stig x, stig y). One training sample consisted of two perturbed images as the model input and the negative aberration vector as the target $\Delta\tilde{F}_i = -\delta_i$.

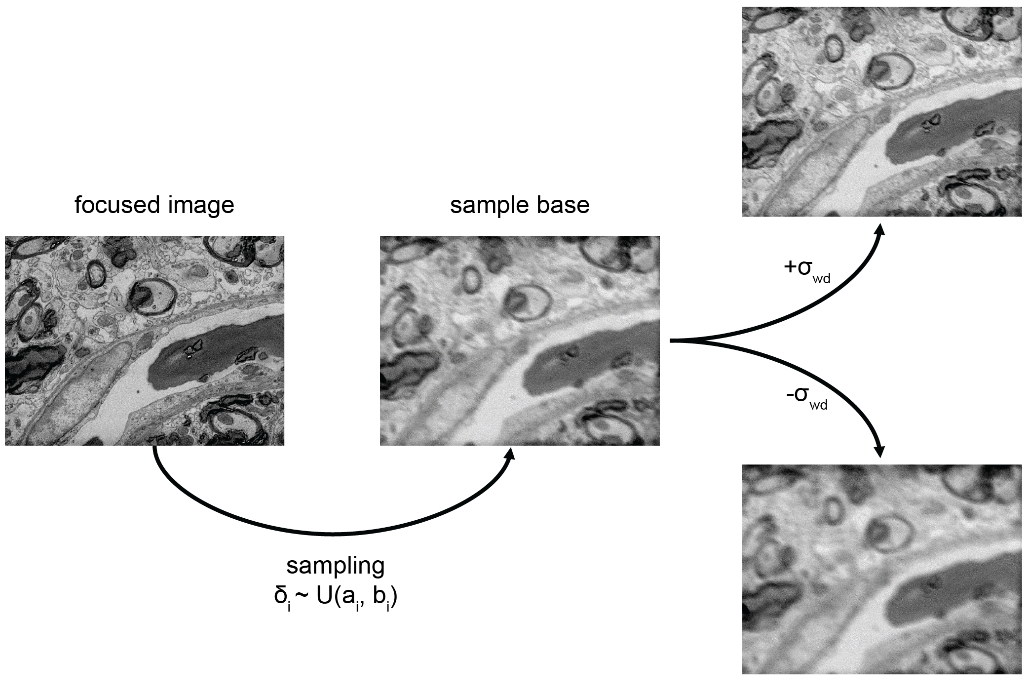


Figure 3.1: Overview of the training data generation. The parameters of a focused beam are changed by adding a uniformly sampled offset δ_i for working distance and stigmator parameters to generate a set of distorted images with known aberration values.

At 23 locations, the aberration vectors were sampled from a working distance range of ± 20 µm and stigmators from ± 0.5. More specifically, every parameter was drawn independently and the resulting aberration vector was created by concatenation. The perturbed (input) images were acquired with a size of 1024×768 pixels and scan speed 3 (2.5 MHz). 17 locations were sampled within ± 20 µm (wd) and ± 5 (stigmators) and the images for these locations were acquired with a size of 2048×1536 pixels and scan speed 1 (10 MHz). The resulting 400 samples were shuffled and split into training (80%, 320 samples) and validation set (20%, 80 samples). The validation set was used to monitor overfitting of the model during training. Test experiments were conducted on different specimen regions and aberration vectors.

Model architectures and training

All models were implemented and trained with PyTorch (Paszke et al. 2019) 1.9.0 and the open-source framework elektronn3[1] using mini batches, L_1 loss (mean absolute error (MAE) between model output and target), step learning rate scheduler (factor of 0.99 every 2000 steps) and AdamW optimizer (Loshchilov and Hutter 2019).

The image-to-scalar architectures used 7 convolutional layers (valid convolution; 3D kernels to share weights across the two inputs using a z-kernel size of 1) followed by 3 fully connected layers (FCLayer). The convolutional layers (Conv3D) were built as follows: convolution, batch normalization, activation (ReLU), max pooling, dropout (Srivastava et al. 2014) with a rate of $p = 0.1$ during training. The architecture for $2 \times 512 \times 512$ inputs consisted of the following layers and parameters (a total of about 1 million trainable parameters):

- Conv3D(input channels: 1, output channels: 20,
  kernel size: (1, 5, 5), pooling size: (1, 2, 2))
- Conv3D(20, 30, (1, 5, 5), (1, 2, 2))
- Conv3D(30, 40, (1, 4, 4), (1, 2, 2))
- Conv3D(40, 50, (1, 4, 4), (1, 2, 2))
- Conv3D(50, 60, (1, 2, 2), (1, 2, 2))
- Conv3D(60, 70, (1, 1, 1), (1, 2, 2))
- Conv3D(70, 70, (1, 1, 1), (1, 1, 1))
- FCLayer(input channels=6860, output channels=250), ReLU
- FCLayer(250, 50), ReLU
- FCLayer(50, 3)

For different input shapes, the parameters of the fully connected layers were adapted as follows:

- $2 \times 128 \times 128$: Linear(140, 100), Linear(100, 50), Linear(50, 3)
- $2 \times 256 \times 256$: Linear(1260, 250), Linear(250, 50), Linear(50, 3)
- $2 \times 384 \times 384$: Linear(3500, 250), Linear(250, 50), Linear(50, 3)

[1]https://github.com/ELEKTRONN/elektronn3

The model output is a correction vector ΔF for working distance (in µm) and stig x and y (arbitrary units). The L_1 loss was calculated without additional weighting as the value range of the different target types (working distance vs. stigmator) appeared sufficiently similar and used a batch size of 8.

To average multiple correction estimates with learned weights, the above architecture was changed to output 4 (3 corrections and an associated score as weight: ΔF_i, s_i), instead of 3 channels. The model was trained by calculating the weighted average of 5 predictions using the softmax (for normalization) of the scores as weights. More specifically, in each training iteration 5 patch pairs were generated from the input and the resulting model output, the weighted average, was compared with the target to calculate the loss.

In the image-to-image case with per-pixel outputs (incl. score; Fig. 3.7), a 3D U-Net was used (Ronneberger et al. 2015) with 3 planar blocks (to allow for shared weights across the two input images), same-convolution mode, which adds zero padding to maintain the input shape, resize convolutions (Odena et al. 2016) for the upsampling, group normalization (Wu and He 2018) (splitting the channels into 8 groups) and 32 start filters. Two subsequent 2D convolution layers followed, which projected the concatenated channels of the two input images down to 4 channels (3 corrections and the associated score) per pixel: Conv2D(input channels=64, output channels=20, kernel size=(1, 1)), activation, Conv2D(20, 4, (1, 1)). In total, the model contained about 0.5 million trainable parameters. A softmax function was applied to the 2D score map output to normalize the score values. The scores were then used to calculate the weighted average of the per-pixel predictions. Multiple dense predictions were combined by calculating their mean.

In both, image-to-scalar and image-to-image, score models a batch size of 4 was used and an additional loss term based on the L_1 loss of the individual (either patch- or pixel-wise) predictions L_1^{ind} was added to the loss of the mean prediction L_1^{mean} ($\alpha = 0.25$):

$$L = (1 - \alpha)\, L_1^{mean} + L_1^{ind} \tag{3.1}$$

The model inputs (grayscale images with intensities between 0 and 255) were rescaled to -1 and 1. Patch pairs, one for each of the perturbed images, were cropped randomly with the same offset. For the model with un-aligned patch offsets in Fig. 3.5a, offset locations were drawn independently for each patch. The following augmentations, implemented in elektronn3, were each applied independently with probability p to the input images I:

- Additive Gaussian noise applied with $p = 0.75$: $I^* = I + X$ with X being a 2D noise map with the same size as I and i.i.d. $X_{i,j} \sim \mathcal{N}(0.0,\, 0.2^2)$.
- Random gamma adjustment: $I^* = I^\gamma$ with $\gamma \sim \mathcal{N}(1.0,\, 0.25^2)$ and $p = 0.75$. Pixel intensities are internally rescaled between 0 and 1.
- Random brightness B and contrast C adaption: $I^* = C(I - I_{mean}) + I_{mean} + B$ with $C \sim \mathcal{N}(1.0,\, 0.25^2)$ and $B \sim \mathcal{N}(0.0,\, 0.25^2)$.

Trainings were stopped after validation loss convergence at $1 \cdot 10^6$ iterations (no-score models), $0.5 \cdot 10^6$ (patch-score model) and $0.2 \cdot 10^6$ (pixel-score model).

Performance tests and MAPFoSt comparison

The convergence properties of the models were assessed by tracking the state of the focus parameters over 10 successive iterations at the same location with a known initial aberration. For each iteration the difference to a focus baseline was plotted for working distance, stigmator x and stigmator y.

Single traces for the DeepFocus model used initial aberrations of $\delta = (30\,\mu\text{m}, -6, 6)$. The parameter baseline was found through manual coarse focus adjustment and three subsequent iterations of the patch-score DeepFocus with 200 ns dwell time, 2048×1536 image size and $20 \times 384 \times 384$ patches, followed by a visual confirmation that a parameter baseline leading to sharp images was obtained.

The image SNRs in Fig. 3.3a,c were calculated with the approach of Sage and Unser 2003 implemented as Imagej plugin and a low-noise image acquired with a 800 ns pixel dwell time as reference. Experiments with the UltraPlus (setup B) in Fig. 3.11 used two iterations of MAPFoSt (400 ns dwell time and $4 \times 786 \times 768$ patches) to adjust the baseline for the unrotated beam scan (Fig. 3.11a) and manual focusing for the 90° rotated scan (Fig. 3.11c,d).

The multi-trace plots were recorded at 9 different locations, regularly spaced on a grid with 80 µm side length. In addition, the mean absolute difference/error (MAE) was calculated each for iteration to estimate the average convergence speed and final variance of the model. The initial focus baseline was found by manual focus adjustment and applying MAPFoSt two times with 200 ns dwell time, resolution of 2048×1536 and 768×768 patches and used as baseline to set the initial aberrations. To address a slight shift in the target focus (working distance) that was observed in the final iterations, likely due to the frequent imaging during the trace acquisition, two iterations of MAPFoSt, or of the patch-score model in the case of Fig. 3.9a, were applied after the trace recording to obtain a more accurate baseline to plot the traces and margins in Fig. 3.9a,b and Fig. 3.10a. Patch locations for DeepFocus were drawn randomly but with a fixed sequence of seeds, i.e. in all traces and at every iteration the same N patch offsets (1 offset per patch pair) were used. Initial aberrations were sampled uniformly within 8 to 12 µm (working distance), -4 to -2 (stig x), 2 to 4 (stig y) with a fixed random seed to ensure an identical distribution of aberrations for MAPFoSt and DeepFocus. The test locations on the specimen of the 9 traces were identical for Fig. 3.9a and Fig. 3.10a.

Error bars were calculated using the uncorrected standard deviation (s.d.) in all plots. All experiments with MAPFoSt were run with the open-source implementation by R. Saxena[2]. The MAPFoSt parameters (including Gaussian approximation of the modulation transfer function, numerical aperture, stigmator rotation and scales) were adjusted by R. Saxena to the used SEM.

[2]https://pypi.org/project/mapfost/4.2.1/

Compute hardware and timings

The model trainings were run on a Windows computer with two Nvidia Quadro RTX 5000 graphics processing units (GPUs), an Intel Xeon Gold 6240 central processing unit (CPU) @ 2.60GHz (36 threads) and 768 GB RAM. Inference was performed directly on the microscope computers (setup A/B) and the time measurements were carried out on the Zeiss Merlin microscope computer (Intel Xeon CPU E5-2609 v2 @ 2.50GHz, 4 threads; 16 GB memory; T1000 GPU) either with the CPU-only or with the CUDA (Compute Unified Device Architecture by Nvidia) backend of PyTorch.

The processing timing started with the perturbed image pair array and ended with a single correction vector, i.e. it included cropping, image normalization, CPU-GPU memory transfers and mean estimation. PyTorch model initialization was not taken into account as it is required only once during startup. Serialized versions of the model were stored and loaded with TorchScript. The MAPFoSt implementation was multithreaded on image patches, i.e. for a 2048×1536 input image and a patch size of 768×768, four parallel processes were spawned. All timings were performed with 2048×1536 images and the relative time comparison was calculated with a cycle time of 0.769 s (corresponding to a pixel dwell time of 200 ns) and computed as the mean of 10 repetitions.

Recalibration procedure

To be able to automatically generate training data on novel setups (DeepFocus recalibration), a separate neural network was designed with the goal of regressing a general and microscope-independent image sharpness score. The model to produce such a score for a single image was based on the image-to-scalar variant of DeepFocus with the following layers:

- Conv3D(1, 20, (1, 3, 3), (1, 2, 2))
- Conv3D(20, 30, (1, 3, 3), (1, 2, 2))
- Conv3D(30, 40, (1, 3, 3), (1, 2, 2))
- Conv3D(40, 50, (1, 3, 3), (1, 2, 2))
- Conv3D(50, 60, (1, 3, 3), (1, 2, 2))
- Conv3D(60, 70, (1, 3, 3),(1, 2, 2))
- Linear(2520, 250), ReLU
- Linear(250, 50), ReLU
- Linear(50, 2)

The model output contained two scores: one for the working distance s_{wd} and one for the stigmation s_{stig} to allow later independent adjustment. The loss was calculated using the L_1 distance between the absolute ground truth targets (working distance, stigmator x, stigmator y) and model outputs. The two, absolute stigmator components of the ground truth were summed before the loss calculation with the model output score s_{stig}. To obtain a single score per image, the minima of N patch predictions (locations selected randomly with fixed initial seed) were calculated independently for each score type (working distance and stigmation) and then summed without additional weights. The resulting single score was used for all experiments.

To turn the so obtainable image sharpness score (objective function) into a microscope-independent autofocus algorithm, it was combined with the downhill simplex method (Nelder and Mead 1965) that minimizes the DeepScore output through iterative adjustment of the focus parameters. For this purpose, the Nelder-Mead Python implementation of F. Chollet[3] was adopted. In case there was no improvement within the last 5 iterations (at most every 5 iterations), the current focus parameters were perturbed with noise drawn from a uniform distribution within ($\pm 2\,\mu\text{m}, \pm 0.5, \pm 0.5$).

Automatic adjustment of the focus parameter at every location was done using the Nelder-Mead-DeepScore autofocus with $10 \times 2 \times 512 \times 512$ patches cropped from an input image with 200 ns pixel dwell time and 2048×1536 pixels. The DeepScore network was trained on the ground truth acquired on setup A (see Ground truth generation). To obtain a threshold to be used as a stopping criterion for the downhill simplex method, the focus was adjusted once manually before starting the procedure and the corresponding sharpness score was evaluated and multiplied by 1.05.

The training image pairs for the DeepFocus recalibration on setup B were acquired on a regular grid with a resolution of 2048×1536 pixels and a dwell time of either 200 ns or 100 ns drawn randomly. The samples of the first 10 locations that had been acquired were used for training, each sampled with 10 aberrations (uniformly drawn between $\pm 20\,\mu\text{m}$, ± 5, ± 5; 100 location-aberration pairs in total; stopping threshold 0.0014). Recalibration was then performed by finetuning the parameters of the last three fully connected layers of a pre-trained DeepFocus model. Finetuning used the training parameters as described for the DeepFocus except for an increased learning rate decay by multiplying with 0.95 every 1000 steps and by limiting training to a maximum of 50,000 steps (approx. 2 h).

[3]https://github.com/fchollet/nelder-mead

3.3 Results

3.3.1 DeepFocus

To obtain a sharp image with a scanning electron microscope, the size of the electron probe must be below the image pixel size. The formation of the electron beam can be controlled by three parameters (working distance: wd, stigmator 1: stig x, stigmator 2: stig y; Fig. 3.2a) within the microscope control software, which effectively adjust the electric current in the objective lens to change the working distance, and two stigmators to correct axial astigmatism. The effect of deviations from optimal parameter values are shown for working distance and one stigmator in Fig. 3.2b.

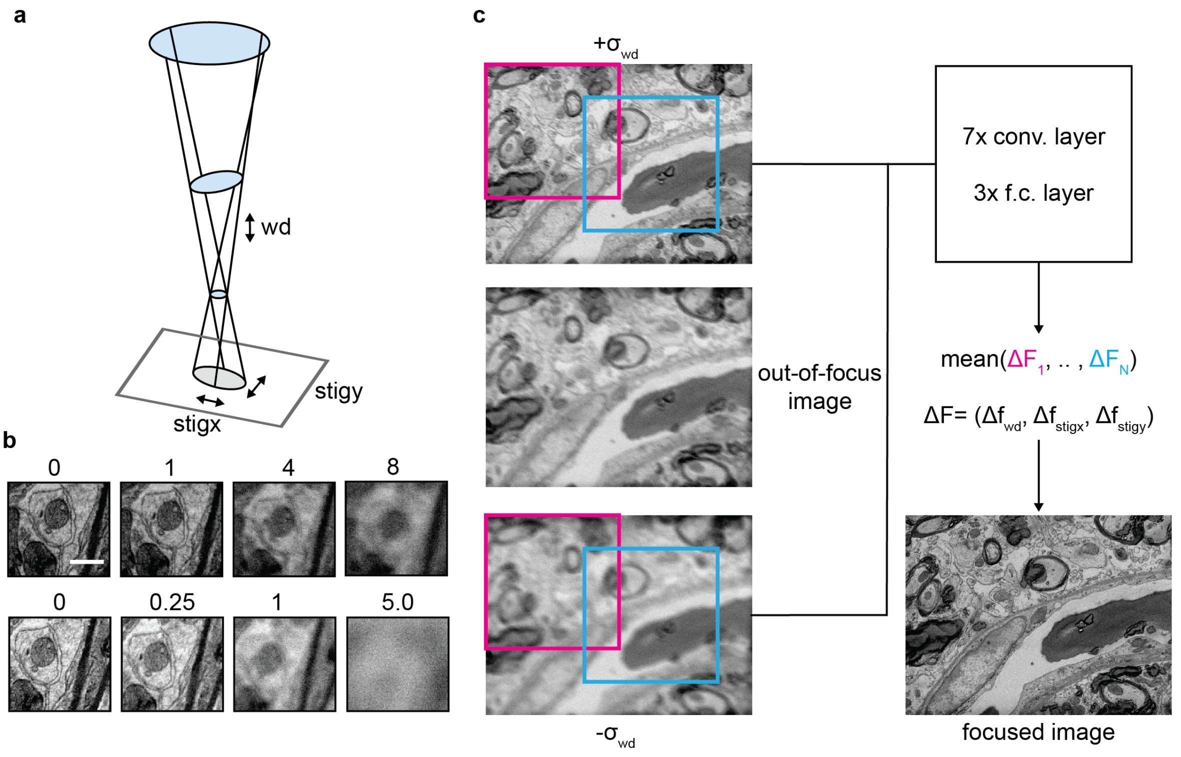


Figure 3.2: SEM beam formation and DeepFocus algorithm. **a** Schematic of the electron beam and the parameters that are controlled by DeepFocus. **b** Defocus - and astigmatism series that shows the influence of mild to severe working distance (top row: 0 to 8 µm) and stigmator deviations (bottom row: 0 to 5 a.u.) on image quality for a brain sample taken with 800 ns pixel dwell time. **c** The out-of-focus image (1024 × 768 pixels) is perturbed (symmetric perturbation $\sigma_{wd} = \pm 5$ µm) and N randomly located patch-pairs of fixed shape are cropped and processed by a stacked CNN (f.c.: fully connected, conv.: convolutional). The mean of N independent predictions is used to calculate a correction term Δf for each focus parameter (wd: working distance, stig x: stigmator x; stig y: stigmator y). All SEM images have 10 nm pixel size. Scale bar in b is 500 nm.

The adjustment of the imaging parameters for week- or even month-long acquisition periods in VEM can hardly be done manually and require automation with little overhead in terms of time and electron dose. Images taken with a working distance shift of the same magnitude but different sign from the focal plane are indistinguishable, which means that it is not possible to infer the correction direction from a single image.

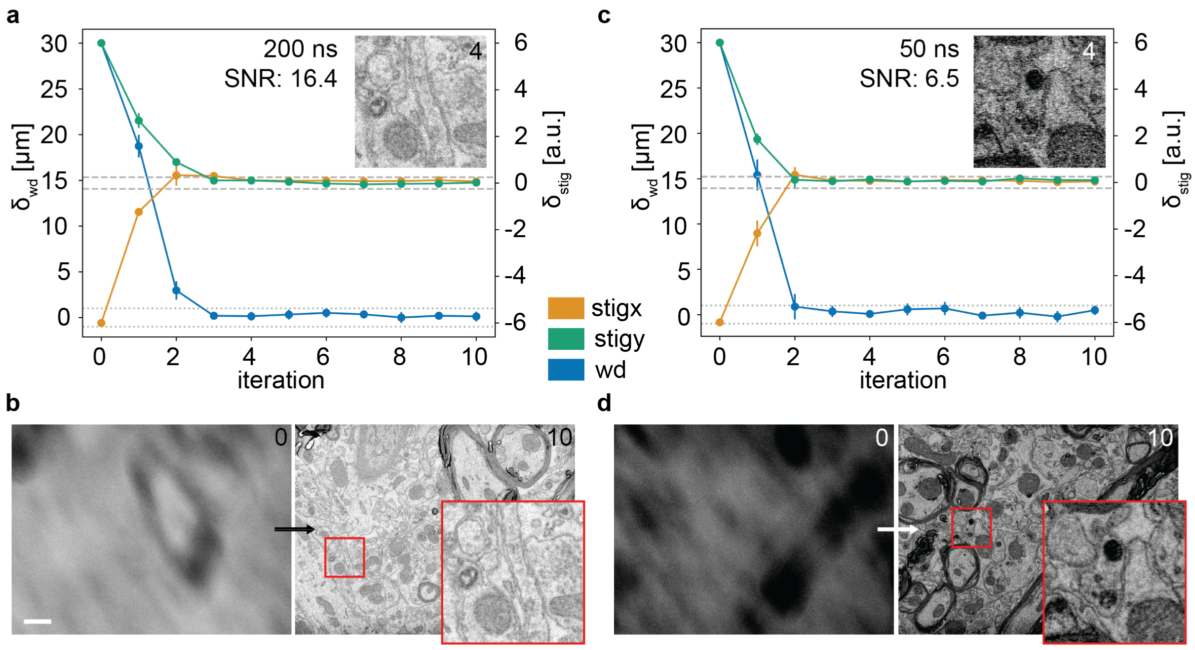


Figure 3.3: DeepFocus convergence properties for different signal-to-noise levels. **a** Convergence plot where each parameter update was calculated as the mean of N predictions with a patch shape of $2 \times H \times W$ (height H and width W in pixels cropped from the two perturbed images) using $5 \times 2 \times 512 \times 512$ input patches and 200 ns pixel dwell time. The Y-axis shows the remaining difference to the initial focus values after each iteration with an initial aberration of 30 µm, +6, -6 (wd, stig x, stig y). Dashed and dotted horizontal lines indicate 0.25 and 1 µm margin of stigmator and working distance, cf. Fig. 3.2b. The perturbed image size was 1024×768 pixels. Numbers in the top right of the example images indicate the iteration count. The plot inset is the same region of interest as in b, but taken at 200 ns dwell time. The image SNR (see Materials and Methods) was calculated relative to the final focus image after iteration 10 (800 ns dwell time). **b** Images acquired with the initial aberrations and after applying DeepFocus. Scale bar is 1 µm. **c**,**d** Same as in a,b but with 50 ns pixel dwell time, including the inset in c. Error bars show the uncorrected standard deviation of the N patch predictions.

Consequently, the DeepFocus algorithm was designed to take subregions (patches) of two SEM images as input, which were acquired with a small working distance perturbation σ_{wd} around the current microscope working distance and stigmator settings

$F = (f_{wd}, f_{stigx}, f_{stigy})$. A pair of patches, one from each perturbed image, is processed by a CNN optimized to infer a correction term leading to a sharp image when added to F. Note, that the offset for a patch pair is the same, e.g. the blue squares (512×512 pixels) in the perturbed images (Fig. 3.2c, top and bottom) have the same relative position.

Multiple subregions are cropped from the perturbed images and processed independently, leading to multiple ΔF_i estimates, one for each input patch pair, of which the mean value is taken as final output for a single iteration:

$$\Delta F = \frac{1}{N}\sum_{i}^{N} \Delta F_i = \frac{1}{N}\begin{pmatrix} \sum_i \Delta f_{i,wd} \\ \sum_i \Delta f_{i,stigx} \\ \sum_i \Delta f_{i,stigy} \end{pmatrix} \tag{3.2}$$

The network was trained until validation loss convergence (about 2 days on a single GPU) and a set of 32 sample locations with different aberration parameters (in total n=320 input image pairs), which refer to deviations from manually adjusted imaging parameters leading to a sharp image. Subsequently, the model was tested on location-aberration pairs that were not part of the training set (Materials and Methods).

The model showed fast convergence towards the target values within three iterations (Fig. 3.3a,b; perturbed image electron dose: $\sim 19\,\text{electron/nm}^2$), even for low-SNR image pairs (Fig. 3.3c,d; electron dose: $\sim 5\,\text{electron/nm}^2$) and was able to extrapolate to large initial aberrations (wd: 30 µm, stig x: +6, stig y: -6) outside the maximal value range covered in the ground truth (wd: -20 µm to 20 µm, stig x: -5 to 5, stig y: -5 to 5; Section 3.2).

The mean estimated correction $\Delta F = (\Delta f_{wd}, \Delta f_{stgix}\, \Delta f_{stigy})$ after one iteration was measured on 9 different locations (regularly spaced grid with edge length 100 µm) for a larger range of initial defocus (working distance perturbation in µm of ±20, ±10, ±5, ±2, ±1) to quantify the goodness of fit of the transformation learned by the model. The relationship between target correction for the working distance $\Delta\tilde{f}_{wd}$ (the negative introduced defocus) and model output Δf_{wd} should ideally be linear (Fig. 3.4a), more specifically it should be $\Delta f_{wd} = c_1 \cdot \Delta\tilde{f}_{wd} + c_2$ with $c_1 = 1$ and $c_2 = 0$. Applying ordinary least squares (OLS)[4] to fit a line resulted in $c_1 = 0.9093 \pm 0.006$ and $c_2 = 0.3436 \pm 0.061$ ($\pm 1\sigma$ interval) which indicates a slight, but significant deviation from the identity function.

Nonetheless, the model successfully learned to infer the direction of the correction. The remaining MAE of the working distance $|\delta_{wd}| = |\Delta f_{wd} - \Delta\tilde{f}_{wd}|$ was closer to the target value $\Delta\tilde{f}_{wd}$ for smaller initial deviations while the initially unchanged stigmator parameters were barely affected (Fig. 3.4b) - both are necessary conditions for convergence.

[4]from the statsmodels Python package (Seabold and Perktold 2010)

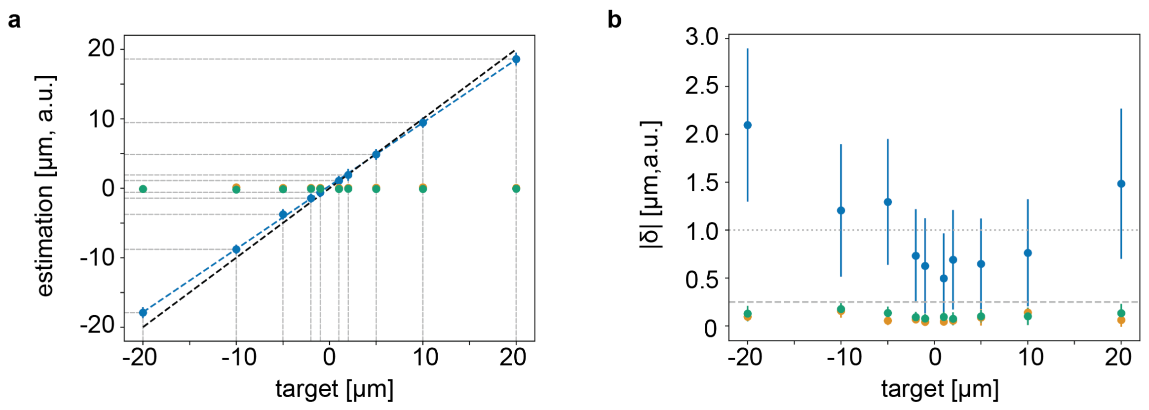


Figure 3.4: DeepFocus single-iteration performance as a function of initial defocus. **a** Correction estimate (mean and s.d. of 9 different locations; in µm for wd and a.u. for stig x and stig y) after one iteration using $5 \times 2 \times 512 \times 512$ input patches with 200 ns pixel dwell time. **b** Remaining mean absolute error $|\delta|$ between estimate and target from a. Colors as in Fig. 3.3a,c. Error bars show the uncorrected standard deviation.

Next, the parameters of the input patch pairs were varied to investigate their effect on the convergence properties. A model that was trained on smaller input patches with 128×128 pixels still showed a stable, but slightly slower convergence (Fig. 3.5a). Proper alignment of the input patch pairs, a strict requirement, e.g. for the algorithm by Binding and Denk (Binding et al. 2013), had only a minor effect even in the extreme case that the patches in an input pair were chosen randomly and independently and could therefore have completely different image content (Fig. 3.5b).

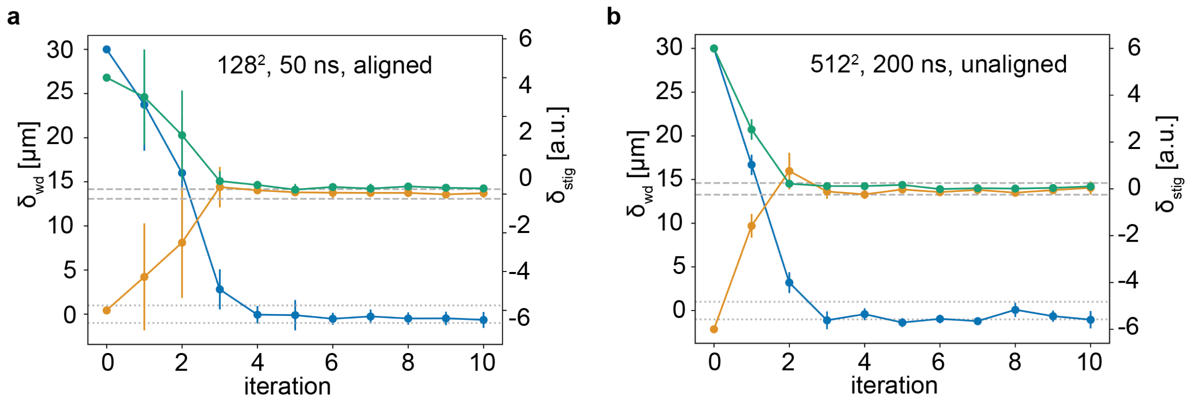


Figure 3.5: Convergence of DeepFocus using different input properties. **a** $20 \times 2 \times 128 \times 128$ input crops and 50 ns pixel dwell time **b** $5 \times 2 \times 512 \times 512$ unaligned input crops, 200 ns pixel dwell time. Crop locations were drawn independently for each perturbed image.

During the development it became apparent how important it is to identify regions with little usable information for an autofocus algorithm. Blood vessels in tissue, for example, can extend over multiple input patches, containing only a flat signal from blank epoxy resin. As a result, the algorithm has no structural information to deduce the defocus and astigmatism which leads to unstable results (Fig. 3.6a,b).

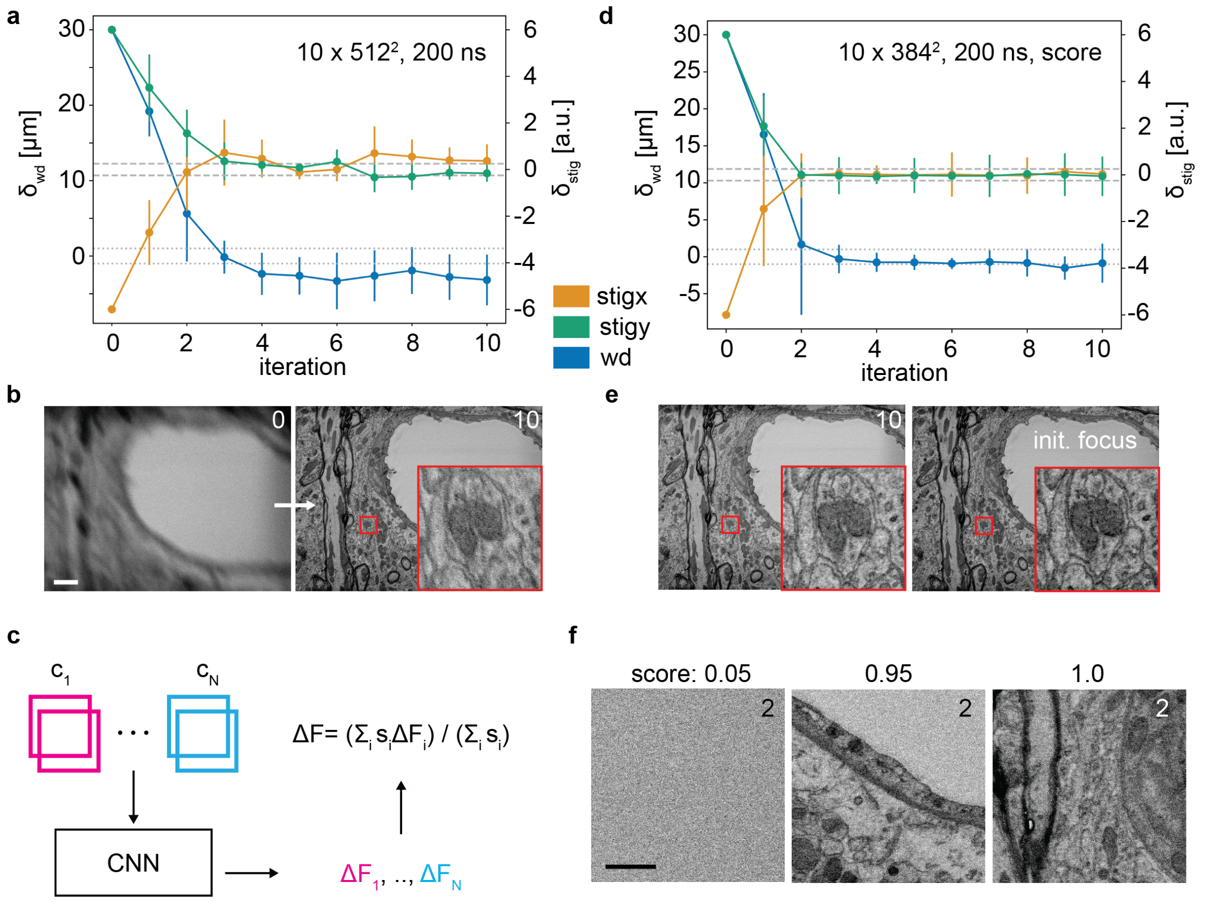


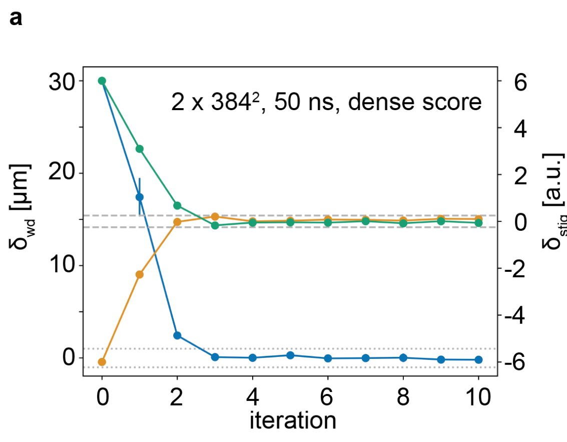
Figure 3.6: DeepFocus model with additional image region score prediction used to calculate a weighted estimator for the aberration correction. **a** Convergence of the model from Fig. 3.3a (using 10 instead of 5 patches) at the location shown in **b** which in large part contains a blood vessel. The perturbed images were acquired with 200 ns pixel dwell time and a resolution of 2048×1568. **c** Model architecture that predicts an additional per patch pair score s_i. **d** Convergence of the patch-score model with $10 \times 2 \times 384 \times 384$ input patches at the same location and settings as in a. Error bars show the unweighted, uncorrected standard deviation. **e** Resulting image using the score model in d and the focused image at the baseline parameters. **f** Example score values for patches used in iteration 2 (fraction of maximum value; original values: 0.0065, 0.1235, 0.1295) together with one of the two input patches. Scale bars are 2 μm in b and 0.5 μm in f.

In an ideal case, corrections estimated from these areas should have no impact on the final correction ΔF. Instead of using custom filtering, the network architecture and the training loss term was modified so that the network outputs an importance weight s_i for every correction estimate ΔF_i, learned in an end-to-end fashion (Fig. 3.6c). The final correction for one iteration is computed as the weighted average of the individual estimates:

$$\Delta F = \frac{1}{\sum_i s_i} \sum_i s_i \Delta F_i = \frac{1}{\sum_i s_i} \begin{pmatrix} \sum_i s_i \cdot \Delta f_{i,wd} \\ \sum_i s_i \cdot \Delta f_{i,stigx} \\ \sum_i s_i \cdot \Delta f_{i,stigy} \end{pmatrix} \tag{3.3}$$

The integration of the weighted average calculation into the model allows to adjust the score outputs with backpropagation during training. At the same time, it requires no additional ground truth targets other than the previously used true corrections $\Delta\tilde{F}$.

The score-extended output was tested on two resolution granularities. First, on the level of patch pairs, where multiple patch pairs from random locations were chosen from the two perturbed images (Fig. 3.6d-f) for the calculation of the weighted average. The convergence trace was acquired using the same setting as in Fig. 3.6a,b but resulted in a more stable convergence and a better final result. A closer inspection of the patch scores suggests a non-linear dependency on the amount of captured structures with large score values for patch pairs that contain at least some structural information and values close to zero for uninformative inputs (Fig. 3.6f).



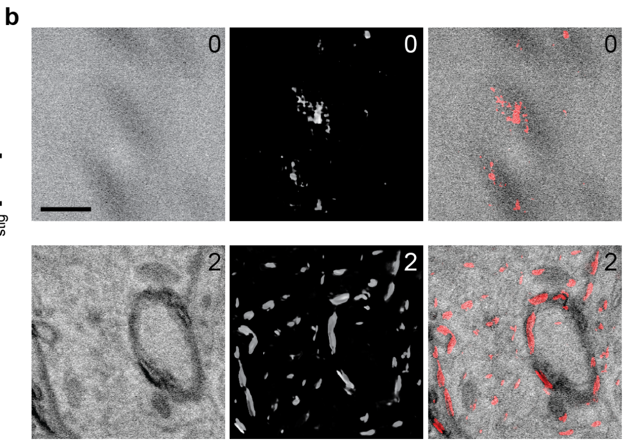


Figure 3.7: DeepFocus model with additional per-pixel score prediction that was used to calculate a weighted average estimator for the focus correction. **a** Convergence of DeepFocus with pixel-wise score predictions using $2 \times 2 \times 384 \times 384$ input crops at 50 ns pixel dwell time and an input image resolution of 2048×1536. **b** Score map of one example patch used in a. The right column shows the composite images of the example input patch (left column) and the corresponding pixel scores (center column) in red at iteration 0 and 2. Scale bar in b is 0.5 µm.

As second output type, predictions were tested at the level of individual pixels, where a U-Net (Ronneberger et al. 2015) learned to output a score and correction map with the same x-y extent as the input patches. This approach showed a fast and stable convergence also with the lowest possible dwell time (Fig. 3.7a). Output scores with large values tended to co-locate with well visible edges in the pixel intensity landscape of the input patches (Fig. 3.7b).

Both the pixel- and the patch-score approaches were more robust toward specimen regions with little contrast information, illustrating that DeepFocus does not require additional, conventional image processing to address such edge cases. Regions which contain little to no structural information and low contrast, which would otherwise make the resulting correction volatile, undergo a soft filtering by assignment of a low weight during inference.

As discussed beforehand, and in addition to the robust correction of image aberrations, a relevant criterion for the actual application of a well-performing autofocus algorithm is that it adds little computational overhead to image acquisition. The DeepFocus processing was therefore timed on GPU and CPU directly on the microscope computer (Fig. 3.8) and compared to the image acquisition time. Inference on GPU outperformed CPU-only processing by about an order of magnitude, which was especially prominent for larger input patches as the preprocessing overhead (Materials and Methods) dominated for small ones.

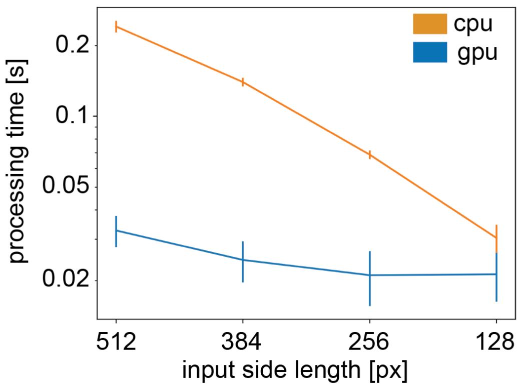


Figure 3.8: AF processing time (mean and s.d. of 10 repetitions and 10 input patches) per input patch-pair of the two input images (2 x 769 ms at 2048 × 1536 pixels, 200 ns dwell time) for different input patch side lengths on the microscope PC. Error bars show the uncorrected standard deviation.

Importantly, with 2.1% of the acquisition time per patch pair, DeepFocus did not add substantial overhead (processing time per 2 × 512 × 512 input: 0.032 s ± 0.004 s, total imaging time for two 2048 × 1536 images: $2 \cdot 0.769\,\mathrm{s} = 1.538\,\mathrm{s}$). For the CPU-only mode, the processing time was found to be 15.6% of the acquisition time (time per patch pair 0.240 s ± 0.011 s), which potentially allows widespread deployment to standard microscope computers even without adding low-powered GPUs. The patch-score model (Fig. 3.6d) with its additional outputs only slightly increased the processing time per patch pair to 0.027 s ± 0.004 s, compared to 0.024 s ± 0.004 s of the baseline model with $2 \times 384 \times 384$ input patches.

Finally, a direct comparison was performed between DeepFocus and MAPFoSt[5] (Binding et al. 2013), which is state of the art for automatic aberration correction in SEM. MAPFoSt uses two test images and models the aberrations with a Bayesian approach to optimally process the available image signal, resulting in the output correction vector ΔF.

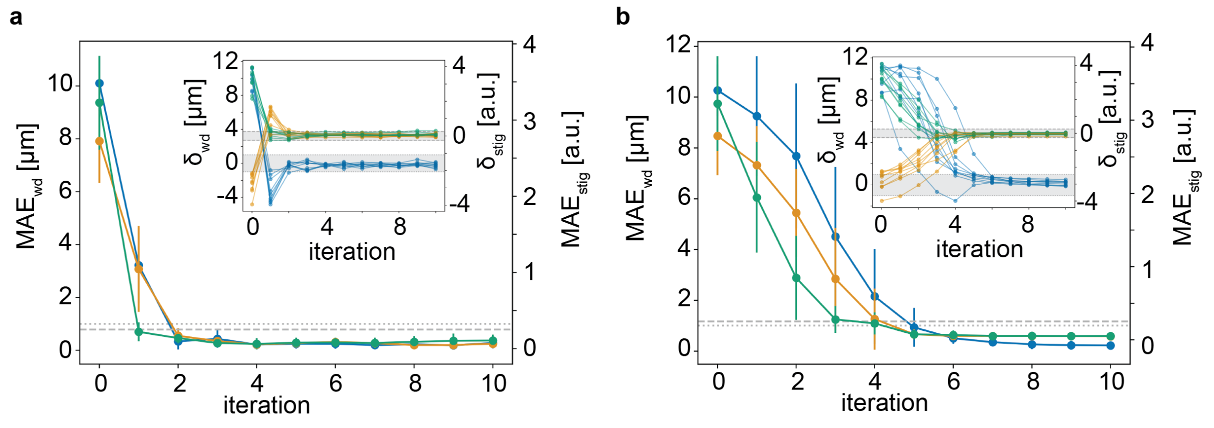


Figure 3.9: MAE of nine convergence traces of DeepFocus and MAPFoSt using 2048×1536 input images. The individual traces are shown in the insets. **a** DeepFocus model from Fig. 3.6d with $10 \times 2 \times 384 \times 384$ patches and 50 ns pixel dwell time. **b** MAPFoSt with $4 \times 2 \times 768 \times 768$ patches and 200 ns pixel dwell time. Colors as in Fig. 3.3a. Dashed and dotted horizontal lines indicate 0.25 and 1 µm margin of stigmator and working distance respectively.

As expected, MAPFoSt successfully converged on all tested aberrations (Fig. 3.9a) but required on average 4 more iterations to converge (MAE$_{wd}$ mean and s.d. of DeepFocus after iteration 2: 0.34 µm ± 0.3 µm vs. MAPFoSt after iteration 6: 0.50 µm ± 0.21 µm) despite using 50 ns pixel dwell time for the two perturbed images with DeepFocus and 200 ns for MAPFoSt. The convergence degraded further with lower pixel dwell times and large initial aberrations (Fig. 3.10a,b).

Additionally, MAPFoSt had almost 4-fold longer processing times in comparison to DeepFocus for the same patch size and almost 30-fold longer times when running DeepFocus on a low-power GPU inside the microscope computer (processing time per 512^2 patch-pair with GPU: 0.032 s ± 0.004 s and CPU: 0.240 s ± 0.011 s vs. MAPFoSt with 0.897 s ± 0.024 s for 512^2 patches and 1.673 s ± 0.018 s for 768^2; Materials and Methods).

The MAPFoSt processing times could likely be further reduced with additional code optimization. Similarly, the DeepFocus model could be subject to inference speed opti-

[5]For the comparison the publicly available Python implementation by R. Saxena was used(https://pypi.org/project/mapfost/). See Materials and Methods for details.

mization, for example, by using mixed/half precision or quantization with 8 bit integers, which in some cases can yield speed-ups of almost one order of magnitude (Hubara et al. 2017).

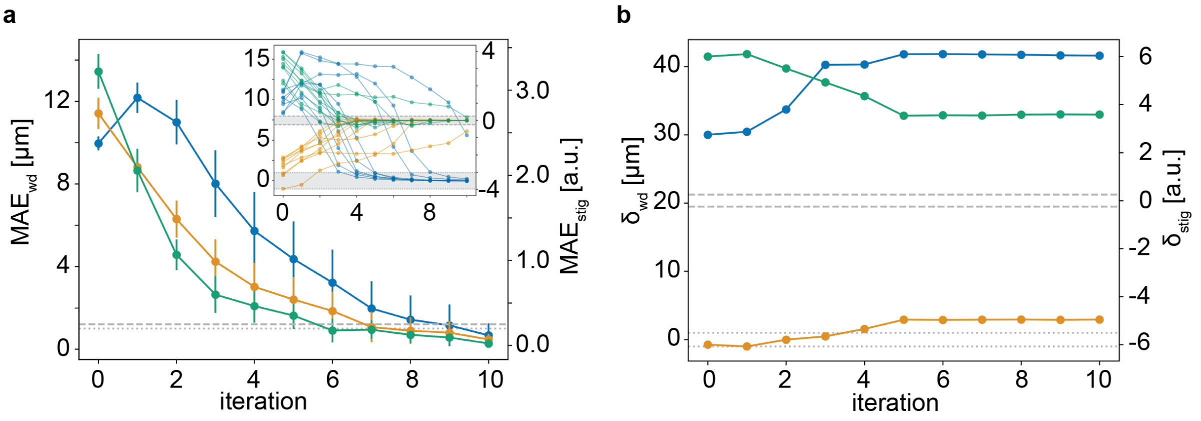


Figure 3.10: MAPFoSt applied to large initial image aberrations using $4 \times 2 \times 768 \times 768$ patches and a perturbed image resolution of 2048×1536. **a** Convergence traces with 50 ns pixel dwell time. Initial aberrations were drawn as in Fig. 3.9. **b** Large initial aberrations (30 µm, -6, 6) and 200 ns pixel dwell time.

3.3.2 Transferability to different settings

Microscope software often contains an existing routine for autofocusing and -stigmation similar to many approaches presented in the past, which, however, perform poorly (Binding et al. 2013). This could be a result of overfitting model parameters or input and processing heuristics of the algorithm to particular test cases. In order to test to which degree the approach suffers from overfitting to its training set, it was evaluated on an unseen resolution specimen (tin on carbon) and on a second microscope setup with different imaging settings.

The transfer to inputs of a novel specimen worked remarkably well (Fig. 3.11a,b) even though the model was only trained on data taken from 32 different locations of a single, biological specimen – an atypically small training set for CNNs. The influence of data augmentation during training was not tested explicitly, but it presumably is crucial for generalization. As expected, when switching to the different microscope setup (setup B; single beam SEM, Zeiss UltraPlus) in combination with modified beam parameters (landing energy, beam current, working distance), the model converged much more slowly or failed to converge entirely with additional rotation of the scan pattern (Fig. 3.11c).

To test whether neural networks could also be used for aberration correction without machine-specific training data or by applying scaled rotation (see Discussion and Binding et al. 2013), an approach coined DeepScore was developed, which should be machine and setting independent, by estimating only the magnitude of the aberration correction $\|\Delta F\|_1$

($\|\cdot\|_1$ being L_1 or Manhatten distance) without direction information from a single image (Materials and Methods, see also H. Kim et al. 2019; H. J. Yang et al. 2020).

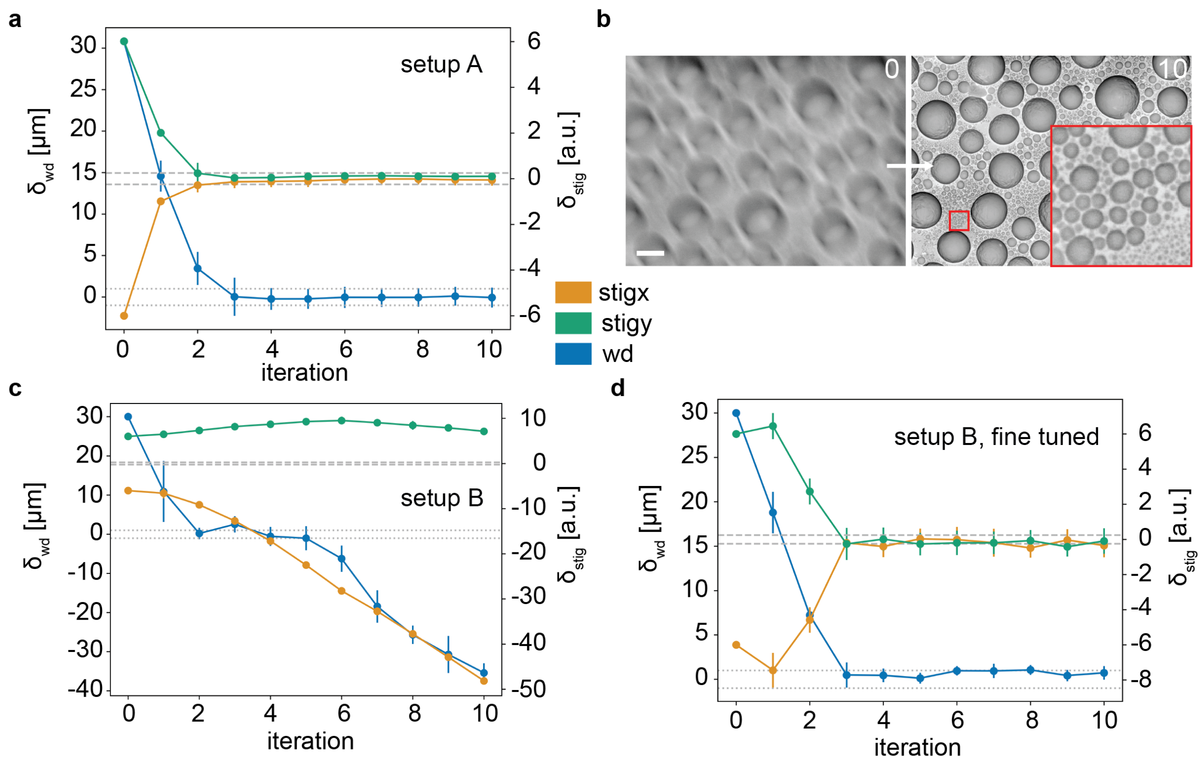
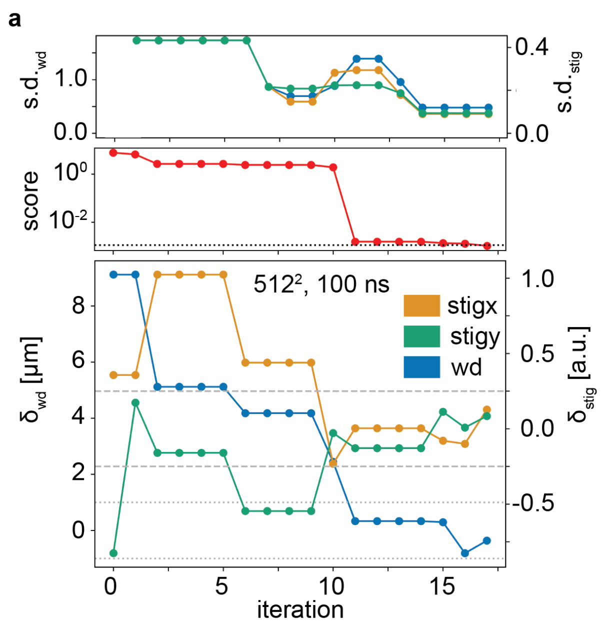


Figure 3.11: DeepScore convergence on an unseen sample and re-calibration to a different setup. **a** Convergence of the model from Fig. 3.6d on a tin on carbon sample (not contained in the training data) on setup A (Materials and Methods) at 100 ns dwell time. **b** Image from a at iterations 0 and 10. Scale bar is 1 µm. **c** The same model as in a applied to tin on carbon on setup B (Materials and Methods). **d** Convergence of the fine-tuned DeepFocus model (last three fully connected layers re-trained) after 50k training iterations on 100 automatically acquired samples at 10 different locations on setup B (Materials and Methods).

The ability to estimate such an image sharpness score ($\|\Delta F\|_1$) combined with an iterative, classical optimization procedure results in an AF algorithm that does not depend on image-pair training data for a specific microscope but rather enables its automatic generation. For this purpose the downhill simplex method developed by Nelder and Mead 1965 was adopted to find minima in the score landscape, which allows to generate new training data almost fully automatically when recalibration of the highly optimized DeepFocus is required.

First, it was tested if the classical optimization with DeepScore is able to infer imaging parameters leading to fine beam probes. For this, the convergence of 14 mild test aberrations, sampled from a uniform distribution (value ranges for wd, stigx, stigy: ±10 µm,



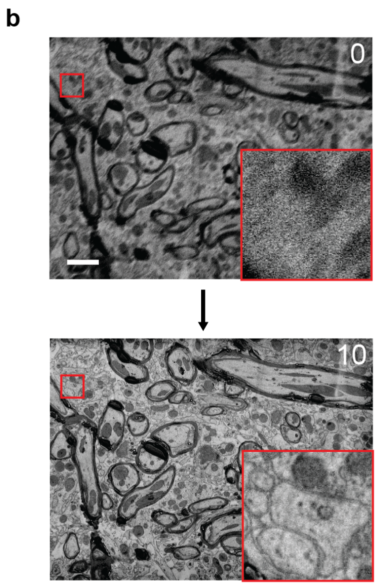


Figure 3.12: DeepScore autofocus convergence analysis. **a** Example convergence trace using the Nelder-Mead optimization of the DeepScore prediction with $5 \times 512 \times 512$ input crops, 100 ns dwell time, input image size of 2048×1568 and a total of 37 score evaluations, i.e. 37 image acquisitions. The s.d. (µm for wd and a.u. for stigs) was calculated from the simplex vertices for each iteration and parameter (iteration 0 is undefined). **b** EM images from the trace shown in a before and after the Nelder-Mead optimization with the DeepScore objective (introduced aberration: 9.11 µm, 0.35, -0.82).

± 1, ± 1), on setup A was recorded. All traces converged, albeit slower than the DeepFocus model (Fig. 3.12). To find the focus baseline, the model from Fig. 3.3a with $N = 10$ patches was used and the stopping threshold for the parameter search was set to the score of the autofocused image[6].

The question remained whether it would be possible to restore the ability of the DeepFocus model to quickly find proper imaging parameters. Therefore, the proposed Nelder-Mead optimization was used in combination with the same DeepScore model (trained on setup A) to create a new minimal training data set on setup B ($n = 10$ locations, 31% of the original training set) with the beam parameters that lead to divergence of the original DeepFocus model (Fig. 3.11c). The fine-tuning (recalibration) on this data set took less than 2 hours (stopped after 50,000 iterations) and resulted in a restoration of the original convergence speed (Fig. 3.11d).

[6]The inferred sharpness score was multiplied by 1.1 and restricted to be ≥ 0.001. The largest so obtained score was 0.0014.

3.4 Discussion

In recent years, deep learning has evolved to state of the art in fields such as natural language processing (Otter et al. 2020) and computer vision (Grigorescu et al. 2020; Kar et al. 2021; X. Liu et al. 2019) and has shown first successes in machine control (Moe et al. 2018; Zeng et al. 2020). Here, it was demonstrated how the problem of autofocusing and autostigmation in SEM can be solved through the use of convolutional networks that iteratively update beam parameters to form a fine electron probe. This data-driven approach showed quick convergence for large initial aberrations and fast scan speeds with down to 5 incident electrons per square nanometer. Without additional optimization, the introduced overhead for imaging and the absolute processing time was small (tens of milliseconds per input) even with direct execution on the CPU of the microscope computer with 4 cores (hundreds of milliseconds).

Random selection of patch locations avoids the introduction of a potentially (more) biased selection heuristic during the model-input preparation with large images. Multiple inputs also allow the design of an architecture that returns a consensus of independently processed inputs. During training, the model output requires a differentiable form, but during inference estimators other than the mean are conceivable, for example, the mean correction of only the N highest scores, or the $\arg\max$.

Instead of random sampling, the patches could also be arranged in a regular grid in the acquired images, or EM scans could be tailored entirely to the height and width of a single patch. For acquisition of a whole volume EM data set, the image size is usually on the order of 5000×5000 pixels or more, which takes 8 times longer to acquire than the images that the timings were compared to here (2048×1536). As the autofocsing procedure requires no minimal image size, as long as it is above the patch shape, it can, e.g., be run on a dedicated, appropriately sized focusing tile, or on continually changing locations in actual stack images, which would distribute the extra electron dosage evenly.

Artificial neural networks are powerful approximators, that in theory, can learn arbitrary mappings between a given input and output (Hornik et al. 1989). It is, therefore, important to constrain the learning problem to achieve a sufficient level of generalization (and computability), in terms of the architecture, like the here used convolutional network, but also to construct reasonable inputs and targets. The target stigmator values, for example, could be transformed into a microscope-independent reference frame by applying a scaled rotation, such as applied by Binding et al. 2013.

Given the large number of adjustable model parameters, the data statistics of the target domain need to be sufficiently covered in the training set and ideally complemented by augmentations. The data set size used for training and validation was untypically small taken at only 40 different specimen locations and a total of 400 input-output pairs (e.g. compared to K. Lee et al. 2021 with more than ten thousand samples). Although augmentations and pooling of patches certainly mitigate this circumstance, enlarging the training set will likely lead to further performance improvements.

The, in the classical sense overparametrized, model outperformed the state-of-the-art approach (Binding et al. 2013) in terms of convergence speed, correction of large aberrations with images acquired at low electron dose, and processing times. Considering how well deep learning approaches perform, particularly in the image domain, this is not surprising since the given problem can be well formulated as a regression of two images into a correction vector of working distance and stigmators.

In addition to the transferability to novel specimen, a procedure was presented to recalibrate the model to new setups or vastly different beam and scan properties almost fully automatically. This data driven approach also allows adjustment to any peculiarities of the used SEM or specimens. Control mechanisms in general might shift from careful, manual fine-tuning of model parameters towards carefully designing the inputs and targets of data-driven models.

Chapter 4

Learning cellular morphology

Parts of this chapter contain text and figures from (Schubert et al. In review) and the peer-reviewed publication (Schubert et al. 2019).

4.1 Introduction

The machine learning toolkit, SyConn, presented in (Dorkenwald, Schubert, et al. 2017) applied CNNs for the detection of mitochondria, vesicle clouds and synaptic junction (further referred to as ultrastructure), and membranes. In addition, SyConn utilized cell skeletons to reconstruct neuron surfaces, which in turn allowed the association of the prior extracted ultrastructure to neurons. The resulting "augmented" neuron representation was the basis for further analysis with random forests (RFs) (Breiman 2001) on a set of hand-designed features. Although highly accurate, this approach required precise, manually traced center lines of neurons in the volumetric image data.

One alternative to hand-designed skeleton features is to learn feature representations using supervised learning directly for a given task, for example using artificial neural networks that perform a semantic segmentation of different cellular compartments on voxel level (H. Li et al. 2020; Macrina et al. 2021). In (H. Li et al. 2020), a 3D extension of the ResNet-18 (He et al. 2016) was applied to a multi-channel input consisting of the segmentation mask of a neuron, the predicted ultrastructure and EM raw data to classify locations into the three major neuronal compartments (axon, dendrite, soma). The excellent performance of this approach comes at the cost of dense voxel inputs. The inference is done on individual segmentation masks, i.e. for each neuron separately, which requires to load the volumetric data of each input channel multiple times. This is a result of organizing and storing volumetric images and segmentation data as small cubes (on the order of hundreds of voxels edge length) and the dense packing of neurites in brain tissue.

Instead of cubes of voxels, cell morphology can also be represented through a surface reconstruction that is computed by a meshing procedure from the cell segmentation, such as marching cubes (Lorensen and Cline 1987). Meshes are a well established and an efficient data structure used in computer graphics that consist of a point cloud (vertices) and edges.

The vertices are connected as polygons (faces or indices), most common are triangles, to form a closed surface area. It should be noted that the mesh data does not contain intensity statistics of voxels that otherwise would be available in the EM image data, but the inclusion of raw EM data has shown to be not particularly relevant for predicting compartments (H. Li et al. 2020). Meshes are the most common way to visualize the neuron segmentation, which means no additional processing is required for methods that can work with meshes directly.

During my master's thesis, I explored the use of light-weight mesh data in the context of VEM connectomics (Schubert 2017), which included a representation of cell reconstructions with multi-views. By training a CNN to classify multi-views it was possible to discriminate between the main three compartments of a neuron: axon, dendrite and soma. Such a multi-view based approach requires less data throughput than a 3D CNN, but needs an intermediate rendering step to generate the view representations. Operating directly on the mesh or point cloud would make this extra step superfluous. Moreover, performing a classification of surface fragments in the case of multi-views, or of voxel cubes in the case of the 3D ResNet, is limited to a sparse coverage of predictions. Since both approaches provide only a scalar output for each input surface or volume the coverage is determined by the anchor points of the inputs. As a consequence, a denser coverage requires denser sampling, which in turn increases redundant overlap. However, studies of synaptic properties usually include spine information (Dorkenwald et al. 2021; Kornfeld et al. 2020) and benefit from automated solutions to detailed compartment prediction.

The EM data sets used in this work (*zebra finch area X, small* and *zebra finch area X, large*; Appendix A) were provided by J. Kornfeld and contain parts of area X – a nucleus in the basal ganglia of the zebra finch song bird that is involved in song learning (Kornfeld 2018). The data sets include a flood-filling neural network (FFN) instance segmentation of neurons and ultrastructure predictions which originated from a collaboration with V. Jain and M. Januszewski at Google Research (see Appendix A for details). The voxel segmentation of cells and ultrastructure was processed to generate a database of neurons, including skeletons, meshes and their associated mitochondria, vesicle clouds and synapses, which is described in Chapter 5. This enriched neuron representation forms the basis for the here presented approaches.

Costa et al. 2016 proposed a method for cell type detection that calculates pairwise similarities between neurons on the basis of their local skeleton geometry. Based on geometric characteristics, such as the location, rate and angle of branches, they performed a hierarchical clustering and were able to identify different neuron types. The inclusion of ultrastructure adds another, informative layer to neuron morphology besides the overall geometry. Additional shape detail, as provided for example by the cell mesh, might further extend the expressiveness of neuron representations.

This chapter illuminates the described identification problems further by presenting machine learning methods that enable their automation to a substantial degree, which in turn lays the basis for a detailed analysis of synaptic wiring. It first introduces multi-view representation of cells and a general approach for point cloud processing based on continuous convolutions (Boulch 2020; Boulch et al. 2020) that avoids rendering overhead

(Section 4.1.2). The results of three applications to morphological analysis of neurons follow, namely an approach for the high-resolution semantic segmentation of compartments (Section 4.3.1), supervised classification of morphologically known cell types (Section 4.3.2) and *a priori* class-wise uninformed clustering (Section 4.3.3).

4.1.1 Multi-view representation of cell reconstructions

Data-efficiency is one key factor towards high-throughput connectomic analysis – ultimately being a prerequisite for petascale data set processing. In this light, the transition from 3D voxel cubes to sparse point clouds via surface meshing appears natural. As a data-efficient alternative for the compartment prediction, one could imagine a 3D CNN architecture that allows inference of voxel cubes in a dense fashion, i.e. returning predictions for every cell fragment in a cube in parallel. In the case of subsequent proofreading, where the agglomeration of supervoxels changes and cells have to be re-processed, the advantage of dense cube processing would disappear since the corrections of neurons is of sparse nature.

Inspired by multi-view CNNs (Qi et al. 2016; Z. Wu et al. 2015) that use 2D projections of entire objects for classification, the here presented cellular morphology neural networks (CMNs) infer various properties directly from cell meshes. Contrary to objects used in the multi-view CNN studies, cells in neural tissue inherit a very fine and elongated structure. This property requires to develop a sampling strategy capable of preserving morphological detail below 100 nm while covering entire neurons that potentially extend over millimeters (Helmstaedter 2013).

Therefore, the view generation (rendering) is performed at densely sampled locations on the neuron. The rendering locations are generated by homogeneously sampling the neuron's surface points (Fig. 4.1a,b), more specifically the mesh vertices are downsampled into voxels with a configurable edge length, which is further referred to as *voxelization*. After the procedure, every occupied voxel is represented by the mean coordinate of the points inside it, or by one contributing element if it is necessary to preserve the original coordinates. The resulting coordinates are then used for the view generation.

Meshes of the different structures (cell, mitochondrion, vesicle clouds and synapses) are stored separately in order to create type-specific feature channels during the rendering (indicated with the different colors, Fig. 4.1c,d). The orthographic projection plane of a single view is model dependent and fixed by a width l_w and height parameter l_h. Multi-view fingerprints are generated from the ultrastructure extended representation (here mitochondria and synaptic junctions) at each rendering location.

The information content of the projections can be optimized by aligning their field of view to the local cell elongation. To find the corresponding rotation, a principal component analysis (PCA) is applied to the vertex coordinates of a local fragment of the cell mesh, which is equivalent to finding the eigenvectors of the covariance matrix of the (centered) coordinates (Deisenroth et al. 2020, Chapter 10).

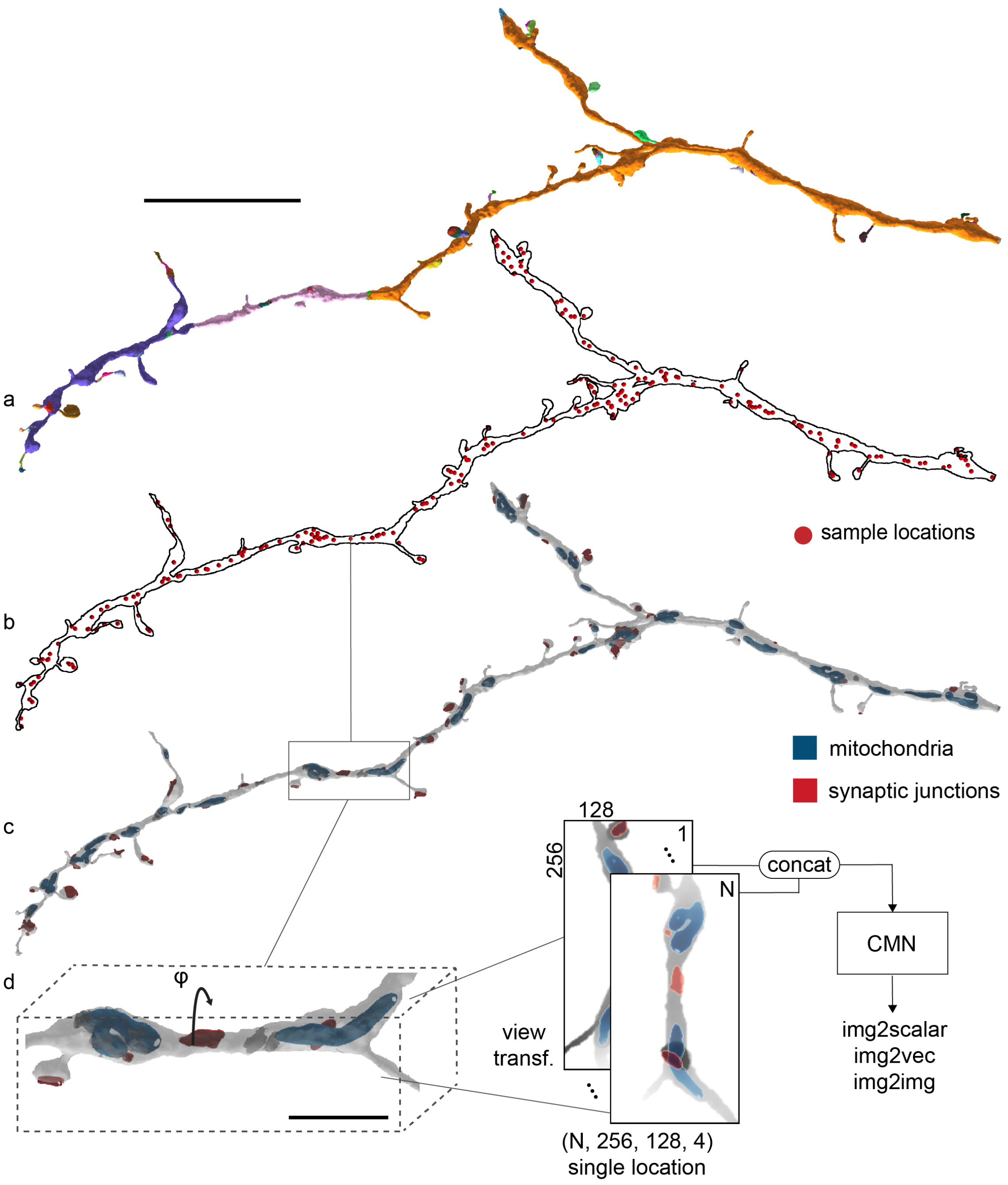


Figure 4.1: Cellular morphology learning networks based on multi-view representation. **a** Agglomeration of supervoxels forming the reconstruction of a dendritic tree. **b** Rendering locations generated with a voxelization into 2 µm voxels. **c** Mesh representation of the neurite including ultrastructure. **d** Local representation of the cell using multi-views (equiangular rotation by φ), generated by orthographic projection from a PCA-aligned subvolume with $l_w = 8\,\mu m$, $l_h = 4\,\mu m$ and a depth of $4\,\mu m$. The two-dimensional views serve as the input for CMNs. Scale bars are 10 µm in a and 2 µm in d. The figure was adapted from (Schubert et al. 2019).

Before rendering, the mesh vertices are centered at the rendering location and rotated into the coordinate system defined by the principal axes, or eigenvectors. Multiple views are rendered by equiangular rotation around the main axis (the eigenvector with the largest eigenvalue) as orthographic projections with a resolution of $N_{pix,x} \times N_{pix,y}$ pixels. Depth is encoded via pixel intensities, whereas higher values represent larger distances, with a minimum and maximum value range of $\pm 2\,\mu\mathrm{m}$ rescaled to 8 bit unsigned integer (0 to 255). Object surfaces outside this range are clipped.

In total, a configurable number of N views with N_{ch} channels, one for each mesh type, is generated for all locations N_{loc}, yielding an array that contains the cell's multi-view fingerprint with shape $(N_{loc}, N, N_{ch}, N_{pix,y}, N_{pix,y})$. The rendering is implemented with PyOpenGL[1] code and supports software-based offscreen rendering (OSMesa backend) and GPU accelerated rendering (EGL).

The described data representation can serve as input for a variety of classification and regression problems (Fig. 4.2) with $k \in [1, \ldots, N_{loc}]$ being the location and $n \in [1, \ldots, N]$ the view index:

- Single image transformation of the n^{th} view at location k into a latent vector $\bar{z}_{k,n}$ to generate embeddings of the local morphology.
- Classification of a single multi-view at location k, resulting in a probability vector $\bar{p}_k$, e.g. for astrocyte and low-resolution compartment detection.
- Pooling a set of views at $N_{loc} \cdot N \geq M \geq 1$ (random) locations for the classification of cell types.
- Pixelwise semantic segmentation of a single view $I_{k,n}$, assigning every pixel a class probability vector $\overline{p}_{x,y}$.

The semantic segmentation of neuron surfaces is subject of Section 4.3.1.

[1]http://pyopengl.sourceforge.net/

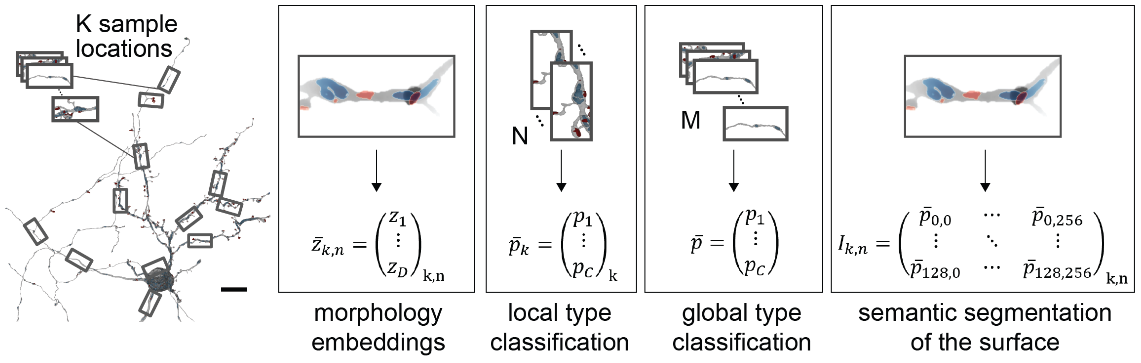

K sample locations
N
M
$\bar{z}_{k,n} = \begin{pmatrix} z_1 \\ \vdots \\ z_D \end{pmatrix}_{k,n}$
$\bar{p}_k = \begin{pmatrix} p_1 \\ \vdots \\ p_C \end{pmatrix}_k$
$\bar{p} = \begin{pmatrix} p_1 \\ \vdots \\ p_C \end{pmatrix}$
$l_{k,n} = \begin{pmatrix} \bar{p}_{0,0} & \cdots & \bar{p}_{0,256} \\ \vdots & \ddots & \vdots \\ \bar{p}_{128,0} & \cdots & \bar{p}_{128,256} \end{pmatrix}_{k,n}$
morphology embeddings
local type classification
global type classification
semantic segmentation of the surface

Figure 4.2: Learning problems that can be addressed with CMNs. The multi-view fingerprint of a cell reconstruction serves as input for a variety of classification and regression problems. Depending on the task, the input is either a single (index n) or the entire N projections of a multi-view at location k, or a random set of M views drawn from all locations K. Scale bar is 10 µm. Figure adapted from (Schubert et al. 2019).

4.1.2 Point-cloud processing using continuous convolutions

Rather then designing a representation that matches the input properties of common models such as discrete convolutions, it can be beneficial to adapt the model architecture to support direct processing of the underlying data. A first attempt of processing point clouds were the voxel-occupancy-based networks that learn to discriminate distributions in a 3D voxel grid (Su et al. 2015). However, these as well have the disadvantages of a fixed-size grid and increasingly large fractions of unoccupied voxels with larger context.

An architecture that processes point clouds should ideally be invariant against translation and the permutation of the input point order and needs to aggregate local and global shape information from point sets (Qi et al. 2017a). The proposed PointNet in (Qi et al. 2017a) addresses permutation invariance by the use of symmetric functions, i.e. fully connected layers with shared weights, which transform 3D point coordinates independently for each point and further uses max pooling to aggregate local features. This pioneering approach however suffered from loss of detail.

Since then, the field has progressed tremendously introducing a multitude of architectures, in particular such that aggregate neighborhood point features hierarchically without explicit spatial kernel, like PointNet++ (Qi et al. 2017b), PointConv (W. Wu et al. 2019), PointCNN (Y. Li et al. 2018) and RandLA-Net (Hu et al. 2020). Other approaches introduce kernel element locations as parameters in addition to the kernel weights, such as Kernel Point Convolution (KPConv) (Thomas et al. 2019), Feature-Kernel Alignment (FKAConv) (Boulch et al. 2020) and ConvPoint (Boulch 2020).

J. Klimesch conducted first experiments on compartment prediction in *zebra finch area X, small* during his bachelor's thesis (Klimesch 2020, supervised by the author of this thesis) with continuous convolutions (Boulch 2020; Boulch et al. 2020). For this purpose, the Python package MorphX[2] was designed and implemented to efficiently generate contiguous point cloud chunks from cell reconstructions by combining the cell mesh vertices with a graph representation, the skeleton of the cell. In this thesis, the point approach for compartment prediction was extended to the large data set (*zebra finch area X, large*) and compared to the method using multi-view semantic segmentation.

The ability to split cell reconstructions into smaller parts is necessary to ensure processability as cell meshes can contain up to millions of vertices - ultrastructure still excluded. In a first step, the closest skeleton node is found for every mesh vertex and the indices (position of the vertex in the global vertex array) of the vertices associated with a node are stored. This mapping can be efficiently computed using a k-d tree (Bentley 1975) ($k = 3$ being the dimensionality of the search space), which in essence creates a Voronoi partitioning of the surface point cloud with respect to the cell skeleton. In addition to the cell vertices, this mapping is provided for every type of ultrastructure (synaptic junctions, vesicle clouds, mitochondria).

A point cloud can be described by the number of points, point density (points per surface area or points per volume) and the extent of the captured neuron or neuron fragment. As these three parameters depend on each other, it is not possible to fix all three of them for varying cell shapes. In order to guarantee a minimum context and a reasonably sized input, the point clouds were defined by two parameters, the context radius r_{ctx} and the number of points.

For the context generation (right half of Fig. 4.3) a node in the cell skeleton graph is chosen and serves as center location for a k-d tree query to collect nodes within a configurable context radius r_{ctx}. The subgraph of the collected nodes is pruned to the connected component that contains the starting node. Collecting all vertex indices from the subgraph nodes (indicated in red) within the radius yields the point cloud chunk by retrieving the corresponding elements from the vertex coordinate array. The four different sources of points (cell, synaptic junction, mitochondrion, vesicle clouds) are converted into a one-hot encoding and used as point features.

Adjusting the density of the source locations for the context generation allows to modify the overlap between adjacent chunks and thereby the degree of redundancy. Each chunk is processed separately by the model and individual chunk predictions are then combined on cell level. For this purpose, multiple predictions of a single vertex are consolidated by majority vote (left half of Fig. 4.3). In contrast to the multi-view approach, the predicted point labels can easily be associated with the mesh vertices by keeping track of the indices during chunking.

A prominent property of U-Nets (Ronneberger et al. 2015) is the symmetric expanding path, which allows seamless processing of images and volumes. The aggregation of learned features in multiple resolution levels enables a large receptive field while being computationally efficient at the same time. Images intrinsically combine positional and intensity information by their grid layout. This structure eases the aggregation of features and therefore transformation of pixel values into the desired output.

[2]https://github.com/StructuralNeurobiologyLab/MorphX

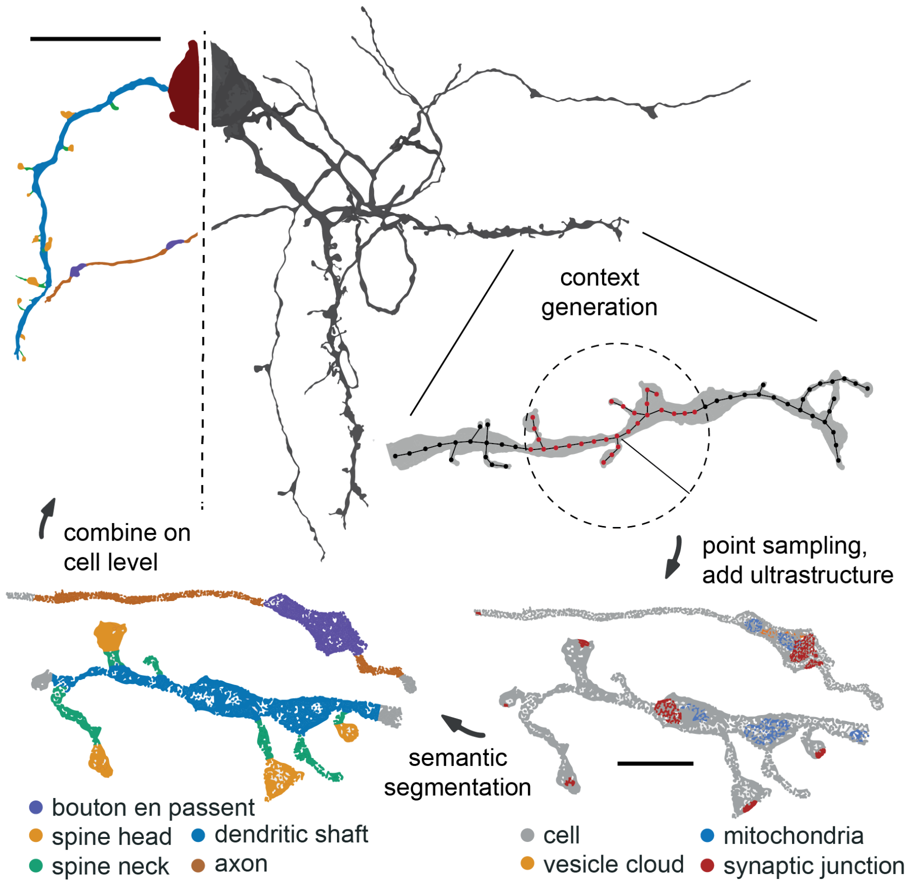


Figure 4.3: Point cloud processing for surface segmentation. Point cloud chunks are retrieved from the neuron and ultrastructure meshes by collecting the node-associated vertices from skeleton subgraphs (indicated in red). The chunks are used as input to the point model and the resulting vertex predictions are combined on cell level. Scale bars are 20 µm for the cell and 2 µm for the point cloud chunk. Figure adapted from (Schubert et al. In review).

The discrete convolution with a kernel $K = \{\mathbf{w}\}$, where $\mathbf{w} \in \mathbb{R}^n$ and $|K|$ being the cardinality, i.e. number of kernel elements and input $X = \{\mathbf{x} \in \mathbb{R}^n\}$ can be written as follows:

$$y = \beta + \sum_{i}^{|K|} \mathbf{x}_i \mathbf{w}_i = \beta + \sum_{i}^{|X|} \sum_{j}^{|K|} \mathbf{x_i} \cdot \mathbf{w}_j \delta_{i,j} \tag{4.1}$$

with β being the bias. The Kronecker delta $\delta_{i,j}$ is added to demonstrate the one-to-one correspondence between image and kernel elements mentioned above. The dimension of the input feature for the first layer could for example be the raw EM images with $\mathbf{x} \in \mathbb{R}^{n=1}$, just consisting of the pixel intensities. Whereas for deeper layers, this becomes $n > 1$, introducing another sum through the scalar product $\mathbf{x}_i \cdot \mathbf{w}_i = \sum_n \mathbf{x}_{i,n} \mathbf{w}_{i,n}$.

Point clouds in contrast are a sparse way of storing shape information and therefore require an explicit representation of spatial location. The reduction of points, analogously to downsampling in images, is performed by hierarchical selection of anchor points for lower resolution levels that aggregate shape features from a large set of points to a smaller one. With every layer, the number of anchor points decreases while the number of (learned) features increases. In the up-path, which agglomerates features from the different resolution levels, usually the same anchor points are used as in the down-path.

The point clouds $\{\mathbf{p}_i\}$ with $\mathbf{p}_i \in \mathbb{R}^3$ obtained from cell and ultrastructure meshes are processed by a continuous convolution operator (Boulch 2020), which parametrizes kernel element locations $\mathbf{c}_i$ in addition to their weights $\mathbf{w_i}$. In contrast to the discrete convolution in Eq. (4.1), the input $X = \{(\mathbf{p}, \mathbf{x})\}$ is convolved with the kernel $K = \{(\mathbf{c}, \mathbf{w})\}$ using an additional weighting term, which is computed by a function ϕ that projects the input points $\mathbf{p}_i$ on the kernel elements:

$$y = \beta + \frac{1}{|X|} \sum_{i}^{|X|} \sum_{j}^{|K|} \mathbf{x_i} \mathbf{w}_j \phi_j(\mathbf{p}_i - \mathbf{c}) \tag{4.2}$$

The function ϕ is realized as a multilayer perceptron (MLP) with three hidden layers (1st layer in- and output channels: $3|K| \to 2|K|$, 2nd layer: $2|K| \to |K|$, 3rd layer: $|K| \to |K|$), which learns to transform the relative distance between input point and all kernel points $\mathbf{p}_i - \mathbf{c}$ into a scalar weight for each kernel element:

$$\phi\colon \mathbb{R}^3 \times \left(\mathbb{R}^3\right)^{|K|} \to \mathbb{R}^{|K|} \tag{4.3}$$

The initial kernel point locations are drawn from a unit sphere and treated as adjustable parameters during training, weights are initialized using Glorot initialization (Glorot and Bengio 2010) and the input to the kernel associated with the convolution of a target point $\mathbf{p}_t$ are its K nearest neighbors, centered at $\mathbf{p}_t$.

The continuous convolution operator is flexible in terms of input and output points, i.e. it does not require the input to be the same points as the output. This means it can be applied in the down- and upsampling path of a U-Net-like architecture. Another use-case, albeit not tested here, is to learn a transformation from surface point clouds to a much sparser target cloud, e.g. the skeleton nodes.

4.2 Materials and Methods

Ground truth generation

The ground truth (Appendix B) for all supervised learning approaches was generated with the help of KNOSSOS[3], a viewing and annotation tool for 3D image data. For the annotation, the skeleton and mesh of a cell were visualized, and in the case of compartment segmentation, the nodes of the skeleton were annotated. The other ground truth data sets either required only visualization of cells (cell type classification), or used the file format mentioned above to store annotated properties of inspected structures.

The compartment annotation for *semseg-fine-train*, *semseg-fine-test-vertices*, *semseg-coarse-train* and *semseg-coarse-test* was performed sparsely to save annotation time, only targeting nodes between changing compartment types. In a subsequent step, all unlabeled nodes received the label of the closest labeled node, the first encounter using a breadth-first search (BFS). This approach allowed to only annotate the compartment boundaries in the skeleton graph and fill-in all intermediate nodes automatically.

Skeleton node labels were propagated to the mesh vertices using Voronoi partitioning, i.e. every vertex received the label of its closest skeleton node (Euclidean distance in nanometer coordinates).

The above sparse labeling scheme still required to annotate entire cells. In order to cover most of the neuron diversity, including also very large cells, a label was introduced that allowed partial annotation of cells by indicating unlabeled subgraphs. As vertices receive the label of the closest annotated node, this indicator was placed with sufficient distance to unlabeled, changing compartment types (e.g. a spine or bouton) to prevent missing labels in extracted contexts. This scheme was applied to generate the data sets *semseg-large-train* and *semseg-large-test* from the *zebra finch area X, large* EM volume. Soma boundary nodes, indicating the start of a soma, were expanded by propagating the soma label up to 40 nodes into the soma using a BFS on the test set.

Multi-view models for semantic segmentation

For the semantic segmentation of cell surfaces, a fully convolutional network (FCN) (Shelhamer et al. 2017) architecture with a "VGG" (Simonyan and Zisserman 2015) backbone was used (configuration B with 10 conv. and 3 fully connected layers, no batch normalization)[4]. The fully connected layers in the backbone are omitted for the use with FCNs. The rational behind the "VGG" network is to use many, small 3×3 kernels. It consisted of 5 blocks (5 resolution levels) each with 2 convolution layers and a 2×2 max pooling (reducing the resolution by a factor of 2 in every dimension) with the following number of output channels (max pooling denoted as 'M'): 64, 64, 'M', 128, 128, 'M', 256, 256, 'M', 512, 512, 'M', 512, 512, 'M'. The convolution was executed in "same" mode, i.e. using zero-padding of 1 pixel on every side to preserve the height and width of the inputs.

[3]https://knossos.app/

[4]Pytorch implementation adapted from https://github.com/pochih/FCN-pytorch by P.C. Huang.

The lower resolution feature maps of the FCN backbone are upsampled by a transposed convolution and concatenated subsequently with higher resolutions by adding skip connections at all downsampling levels (32, 16, 8, 4, 2). For example, the output after the 2×2 max pooling of the first "VGG" output block is connected to the subsequent "VGG" block and additionally combined via skip connection with the input (same resolution, 2-fold downsampled) of the last transposed convolution layer (upsamples to the original resolution) using pixelwise summation. A final convolution layer in the FCN performs the pixelwise classification into the number of output classes N_C using a 1×1 kernel. In summary, the following number of output channels were used for the 5 transposed and 1 standard convolution layer: 512, 256, 128, 64, 32, N_C. A ReLU activation function was applied after every convolution or transposed convolution operation. The model contained 13.3 million trainable parameters.

The input to the model for the dendritic surface segmentation (dendritic shaft, spine neck, spine head, other (axon/soma); *semseg-fine-train*, Appendix B) was set to 256×128 pixels covering a context of $8 \times 4\,\mu\text{m}^2$ with a depth of $4\,\mu\text{m}$, which is sufficient to cover morphologically relevant structures such as spines. In addition, a rather small field of view has the advantage to prevent occlusions, e.g. if another cell process comes close, and a rectangular shape reduces uninformative background.

For the training, mini-batching, the AMSGrad optimizer (Reddi et al. 2018) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate with step decay was used. The Lovász loss (Berman et al. 2018) was chosen to take class balances into account. The hyperparameters for the training are summarized in Table 4.1. All models were trained until visually-confirmed training loss convergence.

Table 4.1: Training parameters for semantic segmentation of dendrites with multi-views. Parameters of the learning rate schedule in the bracket are step frequency and decay factor.

Optimizer	Batch size	init. LR	LR schedule	Loss
AMSGrad	20	0.004	Stepwise (500, 0.99)	Lovász

The model architecture for the coarse segmentation (dendrite, axon, soma, bouton *en-passant*, terminal bouton) remained unchanged except for the input properties, which were adapted to fit a larger context with $40.96 \times 20.48\,\mu\text{m}^2$ at a slightly larger pixel size using 1024×512 during multi-view rendering. The depth extent was set to $40.96\,\mu\text{m}$ in order to reduce clipping artifacts. The training used dice loss (Sudre et al. 2017) to be more sensitive to infrequent classes and hyperparameters are summarized in Table 4.2. The training data set consisted of 45 cell reconstructions (*semseg-coarse-train*, Appendix B).

Rendering locations ($2\,\mu\text{m}$ voxelization of the cell mesh vertices for fine and $40.96/6\,\mu\text{m}$ for coarse segmentation) that were within a $2\,\mu\text{m}$ radius of a manually annotated skeleton node were chosen as context location for the multi-view generation. The generated morphology, index and label views were stored on disk to enable fast data loading during training. To increase data variability, x- and y-axis of the morphology and label view were

Table 4.2: Training parameters for coarse semantic segmentation with multi-views. Schedule parameters in the bracket are step frequency and decay factor.

Optimizer	Batch size	init. LR	LR schedule	Loss
AMSGrad	4	0.0012	Stepwise (500, 0.995)	Dice

flipped during training, each independently with probability $p = 0.5$. Flipping was always applied to both views in order to preserve pixel correspondence.

During inference, rendering locations were generated by downsampling the cell mesh vertices into voxels with edge lengths of 1/6 of the main axis of the context window, i.e. 8/6 µm and 40.96/6 µm for fine and coarse level, respectively. The depth was set to the width of the views L_w.

Voxelization was performed with the `voxel_down_sample` method of the open3D Python package (Zhou et al. 2018). For the alignment of views with the local cell geometry, a PCA was applied on the subset of cell surface points inside a cube with edge length of 8 µm centered at the rendering location.

Point cloud augmentations

Point cloud chunks for the training of the point models were transformed by multiple augmentations. These consisted of random noise added to the point positions, random rotations and flipping, elastic transformations and anisotropic scaling. All point cloud processing methods were implemented in the MorphX package.

The locations of the model input were centered and in some cases scaled by division of 10% of the context radius to rescale values to a reasonable range and independent of the input, without skewing the points to a fix value range. Due to the adaptive kernel element locations, the models were not sensitive to this form of normalization and rescaling was omitted for all other experiments.

The additive noise was drawn from a normal distribution for every point independently $X \sim \mathcal{N}(0, \sigma^2)$ and added to the point coordinates. In order to modulate the noise, the standard deviation was sampled from $\sigma \sim \mathcal{N}(0, \tilde{\sigma}^2)$ with $\tilde{\sigma}$ being the variance of the noise that can be adjusted.

A random rotation of the input cloud was performed with Euler angles (extrinsic xyz) drawn uniformly within an adjustable range. As a side effect, this led to oversampling in the pole regions when drawing from the full angle ranges. After the random rotation, flipping of every axis with was applied with probability $p = 0.5$. The rotation was performed after centering.

The elastic distortions were inspired by (Simard et al. 2003) and adapted to sparse points. Therefore, a three dimensional displacement field with a fixed grid resolution (number of voxels) was generated, and for every voxel, a distortion was drawn from a uniform distribution with a configurable value range. The displacement cube was smoothed using a Gaussian filter with adjustable σ. The distortion cube was interpolated along the full range of the input point cloud and added to the vertex locations.

For the anisotropic scaling every spatial dimension was stretched/skewed independently with a factor $s_{x,y,z} \sim \mathcal{U}(1-r,\, 1+r)$, where r was a parameter to adjust the scaling strength.

In addition, the two input parameters (point count, context radius) were slightly varied during batch generation. Every fourth sample was built with a modified context size, for which a factor was drawn from a normal distribution and multiplied with the base context radius r_{ctx}. This augmentation led to the occasional presentation of small neuron fragments during training, which can be beneficial during inference in the case that not all neurons are segmented perfectly or completely contained in the EM data set. The number of input points was drawn from $\mathcal{U}(1-r,\, 1+r)$.

Point models for semantic segmentation

The architecture of the point models followed a U-Net-like aggregation scheme with a down-path to learn local shape features with increasing abstraction and decreasing resolution and an up-path which agglomerates them across the different resolution levels. The convolution layers in the up-path project the features of a small input point cloud onto a larger output cloud and concatenate them with the point features that were outputted by the convolution in the down-path on the same resolution level.

The kernel size, i.e. the number of kernel weight-locations was 16 for all kernels and kernel input points were normalized to a unit sphere. For the experiments, the architecture implementations provided in the ConvPoint (Boulch 2020) and LigthConvPoint (Boulch et al. 2020) Python packages were used and adapted. The point sampling in the down-path was performed with space quantization as proposed in (Boulch et al. 2020). The latter performs a repeated downsampling with decreasing voxel sizes (factor 0.5) until the desired number of points is reached. Random points are removed if the point cloud cardinality exceeded the target count.

Table 4.3: Training parameters for the surface segmentation with point clouds (*zebra finch area X, large*). Schedule parameters in the bracket are step frequency and decay factor.

Optimizer	Batch size	init. LR	LR schedule	Loss
Adam	4	0.002	Stepwise (100, 0.996)	Cross entropy

For the semantic segmentation of neuron surfaces in *zebra finch area X, large*, the architecture in Table 4.4 (2.3 million trainable parameters) was used for three models dedicated to different sets of classes (axon, dendrite, soma; dendritic shaft, spine neck, spine head; axon, bouton *en-passant*, terminal bouton). The models were trained on the *semseg-large-train*, (Appendix B) data set with the parameters from Table 4.3, Adam optimizer (Kingma and Ba 2015), context radius r_{ctx} of 15 µm and 15,000 input points. Non-uniform class weights were applied in the cross-entropy loss to address the class imbalance in the finer segmentation tasks: axon, bouton *en-passant*, terminal bouton: $[1, 2, 2]$; dendrite, spine neck, spine head: $[1, 2, 2]$. Depending on the task, ground truth labels that were outside of a models output classes were ignored, e.g. soma and bouton labels for the dendrite-spine-neck model. For the axon-dendrite-soma model, finer structures were remapped to the corresponding upper-level semantic class, i.e. spine neck and spine head were treated as dendrite, and the two bouton classes as axon.

The model that was trained for the comparison with the multi-view approach on the small data set ground truth (*semseg-coarse-train*) used the architecture from Table 4.4, training parameters as in Table 4.3, a context radius r_{ctx} of 15 µm, 15,000 input points and 2 times larger weights for the two bouton classes during training.

Table 4.4: Number of input and output channels and point cardinality (-1 refers to the initial size) for semantic segmentation architecture separated into down- (top) and up-path (bottom). Features are concatenated in the up-path between layers with the same output point cardinality.

inp. points	out. points	inp. channels	out. ch.	neighborhood
-1	-1	4	64	16
-1	2048	64	64	16
2048	1024	64	64	16
1024	256	64	64	16
256	64	64	64	16
64	16	64	128	16
16	8	128	128	16
8	16	128	128	4
16	64	256	128	4
64	256	256	64	4
256	1024	128	64	8
1024	2048	128	64	8
2048	-1	128	64	8

For the dendrite segmentation comparison a first model was learned to discriminate the cell into dendrite and axon/soma (training parameters as above but using dice loss; class weights $[2, 1]$) and a second, fine-scale model to infer dendritic shaft, spine neck and head. Cell ultrastructure information was only used for the first model. For the parameter search of the second model, the architecture from Table 4.3 was adapted depending on the input point cloud cardinality to also allow processing of small point clouds. For 512 input points architectures with the following layer specifications were used: (1: 32 channels, 32 neighbors, no reduction), (2: 32, 32, reduction to 256 points), (3: 64, 32, reduction to 64 points), (4: 64, 16, 16), (5: 64, 8, 8), (6: 64, 4 upsampling to 16, residual to 5), (7: 64, 4 upsampling to 64, residual to 4), (8: 32, 8, upsampling to 256, residual to 3), (9: 32, 16, upsampling to original point cloud, residual to 2), (10: fully connected shared

across all points, residual to 1). Two more layers between layer 1 and 2 and layer 8 and 9 respectively were added for 1024 input points: (1/2: 32, 32, reduction to 512), (8/9: 32, 16, upsampling to 512 + residual). For 2048 points two layers (additional to the 1/2, 8/9 layers) were added: (1/2: 32, 32, reduction to 1024), (10/11: 32, 16, upsampling to 1024 + residual). Models with more than 2048 input points shared the same architecture as for 2048 points, but changed the reduction pathway to: no reduction, 2048, 1024, 256, 64, 16, 8. Input points were not rescaled to the unit sphere and the trainings were run with an initial learning rate of 0.001 (scheduler step size of 1000, decay 0.99), batch size of 32 and optimized with Adam. Trainable model parameters were in the range of 0.5-0.6 million, depending on the architecture.
During training, the following input augmentations were applied during training:

- Point-wise additive spatial noise with $X \sim \mathcal{N}(0, \sigma^2)$ and $\sigma \sim \mathcal{N}(0\,\text{nm}, (20\,\text{nm})^2)$
- Centering
- Random rotation around all three spatial dimensions and independent flipping/mirroring of every spatial axis ($p = 0.5$)
- Elastic distortions (grid resolution 40^3 and smoothing of $\sigma = 6$)
- Anisotropic scaling $s_{x,y,z} \sim \mathcal{U}(0.95, 1.05)$
- Number of input points noise with factor $\sim \mathcal{U}(0.9, 1.1)$
- Context size variation applied to every fourth sample with a factor sampled from $\mathcal{N}(1, 0.1^2)$, truncated at $[0.8, 1.2]$

Input point clouds were voxelized to a meshing-procedure independent resolution as a function of their type (80 nm for cell surface and 100 nm for all ultrastructure). Source locations were drawn randomly from all skeleton nodes for the training. During inference they were retrieved by voxelization of the skeleton nodes with $r_{ctx}/5$ for the dendrite segmentation comparison and $r_{ctx}/2$ for all other experiments to ensure a homogeneous coverage.

Point models for cell type classification

The ground truth was split into training and test data using 10-fold cross-validation taking class support into account[5]. Due to limited ground truth, the trainings were performed without validation and evaluated on the test set after seven days of training. Each split was used to train three models, each starting with a different random seed for training batch generation and initial weights to estimate the model variance. The context generation was parameterized with radius and number of points. Seed nodes for context locations were sampled uniformly from cell skeletons. The ground truth (*celltypes-large*, Appendix B)

[5]`StratifiedKFold` method from the scikit-learn Python package

contained 11 classes, which represent putative cell types contained in the EM data set: STN, DA, MSN, LMAN, HVC, TAN, GPe, GPi, FS, LTS, NGF.

The model architecture consisted of 5 ConvPoint layers, each using 16 kernel elements, group normalization (always grouping two channels) before swish activation (Ramachandran et al. 2018) with the following parameters (output channels, reduction to N points, k nearest neighbors): (64, 4096, 32), (128, 1024, 32), (256, 512, 16), (256, 256, 16), (512, 128, 16). The resulting 512 features were averaged across the 128 anchor points. An additional dropout (rate 0.3) was applied before the final two fully connected layers with 128 and $N_C = 11$ output channels. In total, the model consisted of 3.8 million trainable parameters. The reduction was done with the heuristic random sampling (Boulch 2020), that prevents oversampling of the same points.

The Training parameters are summarized in Table 4.5. To speed up the data preparation during training, a single batch (batch size 10) contained random contexts of only one cell. Parameter updates were performed after accumulating gradients of 10 batches to improve the learning signal.

Table 4.5: Training parameters for the supervised cell type point model. The value in the bracket of the batch size is the number of samples presented within each iteration. Schedule parameters in the bracket are step frequency and decay factor.

Optimizer	Batch size	init. LR	LR schedule	Loss
Adam	100 (10)	$5 \cdot 10^{-4}$	Stepwise(100, 0.99)	Cross entropy

The following augmentations were applied to the input during training:

- Point-wise additive spatial noise with $X \sim \mathcal{N}(0, \sigma^2)$ and $\sigma \sim \mathcal{N}(0\,\text{nm}, (40\,\text{nm})^2)$
- Centering and scaling by division of 10% of the context radius
- Random rotation around all three spatial dimensions and independent flipping of every spatial axis ($p = 0.5$)
- Elastic distortions (grid resolution 40^3 and smoothing of $\sigma = 6$)
- Anisotropic scaling $s_{x,y,z} \sim \mathcal{U}(0.9, 1.1)$
- Number of input points noise with factor $\sim \mathcal{U}(0.9, 1.1)$
- Context size noise applied to every fourth sample with factor $\sim \mathcal{N}(0.7, 0.1^2)$, clipped to between $[0.33,)$ to occasionally present small cell fragments.

During inference with a batch size of 20, the final classification was found as the majority vote of multiple predictions to make the prediction more robust. The predictions were performed at N randomly selected locations, drawn from the cell skeleton nodes after voxelization to 2 µm. Input point clouds were voxelized with 70 nm for cells and synapses, and all others with 100 nm.

Multi-view models for cell type classification

The model performance was evaluated using a 10-fold cross-validation (using the same procedure as in Section 4.2), each with three different random seeds, multi-views with a random set of $N_{views} = 20$ views, and random flip augmentation (independently in x and y for each view with probability 0.5). The training parameters are summarized in Table 4.6.

Table 4.6: Training parameters for the multi-view cell type classifier. Schedule parameters in the bracket are step frequency and decay factor.

Optimizer	Batch size	init. LR	LR schedule	Loss
Adam	20	$1 \cdot 10^{-3}$	Stepwise (750, 0.99)	Cross entropy

Cell and ultrastructure meshes were used to render the input views (input channels: 4). In addition, the ratios of predicted symmetric synaptic area over the total synaptic area of the entire cell and dendritic compartments were added as scalar input ($N_{scalar} = 2$) to the first fully connected layer of the network.

Seven convolutional layers (Conv3D) were stacked and each performed, applied in the listed order, 3D convolution (shared weights in z), batch normalization, ReLU activation, max pooling and dropout (rate: 0.08) and three subsequent fully connected layers (FCLayer):

- Conv3D(input channels: 4, out channels: 20,
  kernel size: (1, 5, 5), pooling: (1, 2, 2))
- Conv3D(20, 30, (1, 5, 5), (1, 2, 2))
- Conv3D(30, 40, (1, 4, 4), (1, 2, 2))
- Conv3D(40, 50, (1, 4, 4), (1, 2, 2))
- Conv3D(50, 60, (1, 2, 2), (1, 2, 2))
- Conv3D(60, 70, (1, 1, 1), (1, 2, 2))
- Conv3D(70, 70, (1, 1, 1), (1, 1, 1))
- FCLayer(input channels: $4200 + N_{scalar}$, output channels: 100), ReLU activation
- FCLayer(100, 50), ReLU activation
- FCLayer(50, $N_C = 11$)

The model contained 0.5 million trainable parameters.

Rendering was performed on-the-fly during training with $N = 4$ projections per rendering location, found by 8/3 µm voxelization of the cell mesh vertices. To speed up batching of training samples the rendered views of a cell were cached and re-used up to 200 times to sample a random set of N_{views} views. A batch always contained at least one multi-view from every neuron type to guarantee a diverse learning signal.

During inference (batch size of 10), the final classification of a neuron was found via majority vote of the $\left\lfloor \frac{N_{loc} \cdot N}{N_{views}} \right\rfloor$ predictions, with N_{loc} being the number of rendering locations (8/3 µm voxelization) and $N = 4$ the number of projections per multi-view. If $N_{loc} \cdot N < N_{views}$, views were drawn with replacement.

Point model for cell clustering

For the self-supervised training via triplet loss (Schroff et al. 2015) a model was trained to embed the morphology (cell and ultrastructure) of two spatially nearby locations of cell C_A closer in a 10-dimensional latent space than a cutout of a different cell C_B (drawn randomly). The model architecture was the one used for the supervised cell type classification.

The first context center location (the coordinate of the source node used for context generation, termed context center), was drawn uniformly from all cell skeleton nodes in C_A. The second context center was drawn uniformly within 15 µm distance along the same cell skeleton. This training procedure did not require any additional manual annotations and was performed on sufficiently large neuron reconstructions. Neuron reconstructions that had a bounding box diagonal less than two times the input context of the model (here < 30 µm) were excluded. The training was performed on the *zebra finch area X, large* volume.

The input was set to $r_{ctx} = 15$ µm and 25k points. Training parameters are summarized in Table 4.7. Instead of cross-entropy loss, the following regularized margin ranking loss was used to learn the 10D output of the model:

$$\text{loss}(x_0, x_+, x_-) = \max(0, r_+ - r_- + \alpha) + \frac{\lambda}{3}\left(\|x_0\|_2 + \|x_+\|_2 + \|x_-\|_2\right) \tag{4.4}$$

with $r_{+,-} = \|x_0 - x_{+,-}\|_2$ being the distance between reference and similar/dissimilar location, $\lambda = 10^{-6}$ a factor for the regularization term to prevent latent vectors from diverging and $\alpha = 0.2$ a minimum margin.

Table 4.7: Training parameters for morphology embedding model. Schedule parameters in the bracket are step frequency and decay factor.

Optimizer	Batch size	init. LR	LR schedule	Loss
Adam	16	$5 \cdot 10^{-4}$	Stepwise (250, 0.995)	Margin Ranking

The following augmentations were applied to the input during training:

- Point-wise additive spatial noise with $X \sim \mathcal{N}(0,\, \sigma^2)$ and $\sigma \sim \mathcal{N}(0\,\text{nm},\, (40\,\text{nm})^2)$
- Centering with additive spatial noise with $\sim \mathcal{U}(-500\,\text{nm},\, 500\,\text{nm})$ in every spatial dimension. Scaling by division of 10% of the context radius.
- Random rotation around all three spatial dimensions and independent flipping of every spatial axis ($p = 0.5$)
- Elastic distortions (grid resolution 40^3 and smoothing of $\sigma = 6$)
- Anisotropic scaling $s_{x,y,z} \sim \mathcal{U}(0.9,\, 1.1)$
- Number of input points noise with factor $\sim \mathcal{U}(0.9,\, 1.1)$
- Context size noise applied to every fourth sample with factor $\sim \mathcal{N}(0.6,\, 0.1^2)$

Local embeddings, spatially represented by their context center, were aggregated to cell level by calculating their mean within the same compartments (axon, dendrite) and adding the two resulting vectors. During inference, context centers of a cell were generated using voxel downsampling of the mesh vertices with a voxel size of half the context size (7.5 µm). Every cell skeleton node was finally assigned the embedding vector associated with the spatially closest context center. Input point clouds were voxelized as for the cell type classification.

Compute infrastructure

The experiments were executed on the wholebrain cluster hosted by the Max Planck Computing and Data Facility (MPCDF) in Garching, which consisted of 18 compute nodes. Each node was equipped with 2 NVIDIA Quadro RTX 5000, 20 cores (Intel Xeon CPU E5-2660 v3 @ 2.60GHz) and 256 GB of RAM. For the management of compute jobs SLURM (Yoo et al. 2003) was used. All models were implemented and trained with PyTorch and elektronn3 and used the SyConn2 package (Chapter 5) to interface the VEM data.

Performance metrics

The prediction performance of the models was assessed with the F_1 (harmonic mean of precision and recall) and accuracy. If not stated otherwise, reported F_1-scores are the unweighted averages of per-class F-scores, i.e. not taking into account the class support.

4.3 Results

4.3.1 Compartment prediction

The identification of pre- and postsynaptic sites is based on cues in the EM image data, with the strongest being synaptic vesicles in the presynaptic cell, which contain neurotransmitters that are released by an action potential. This can be either done explicitly by detecting such vesicles and assigning it to its nearby synaptic junction or indirectly by training a CNN to predict pre- and postsynaptic site densely on a voxel level, represented, e.g. as vector field (Buhmann et al. 2021) or as binary masks (Turner et al. 2020). Both approaches solve two problems: Firstly, the identification of pre- and postsynaptic site and, secondly, their assignment to cell reconstruction instances.

These two problems were addressed separately: Based on the sparse mesh representation of cells, consequently allowing a large context, basic functional compartments of neurons namely axon (presynaptic site), dendrite and soma are detected. The assignment of cell partners to a putative synapse object is done by intermediately identifying cell-to-cell contacts, which is the topic of Section 5.3.1.

In (Schubert et al. 2019), an ablation study was conducted for the image-to-scalar classification of multi-views into compartments (axon, dendrite, soma), which showed that the exclusion of depth information and a reduced pixel resolution only marginally affects performance. In contrast, sufficient context (saturated at approx. 8 µm) and the presence of ultrastructure were critical. The input for the semantic segmentation approaches with multi-views and continuous convolutions was designed accordingly, in terms of context and available ultrastructure information.

Multi-view models

Semantic segmentation is the dense classification of every single element of the model's input, e.g. all the pixels of an image. More specifically, every pixel is either assigned a class label directly (hard classifier, e.g. support-vector machine, C. M. Bishop 2006) or a class conditional probability vector (soft classification, such as logistic regression) which usually is converted into the final class label by applying $\arg\max$.

Learning a model to perform semantic segmentation of single views, generated by the multi-view representation, would result in pixel-wise labels, but relating these to the neuron surface is not straightforward. The orthographic projection transforms the three-dimensional object into two-dimensional pixel locations of an image, which is difficult to invert. A procedure similar to (Boulch et al. 2018) was developed and implemented to track the rendered surface representation by generating an additional index map.

To aid mapping back the semantic segmentation of a view to the original surface locations of the cell, a second projection with same camera setting is rendered. The first is the morphology view, which captures the surface of the cell and ultrastructure including the type of mesh (morphology view), while the second contains the index of each cell-mesh vertex or face that contributed to the rendering of the morphology view (Fig. 4.4a) en-

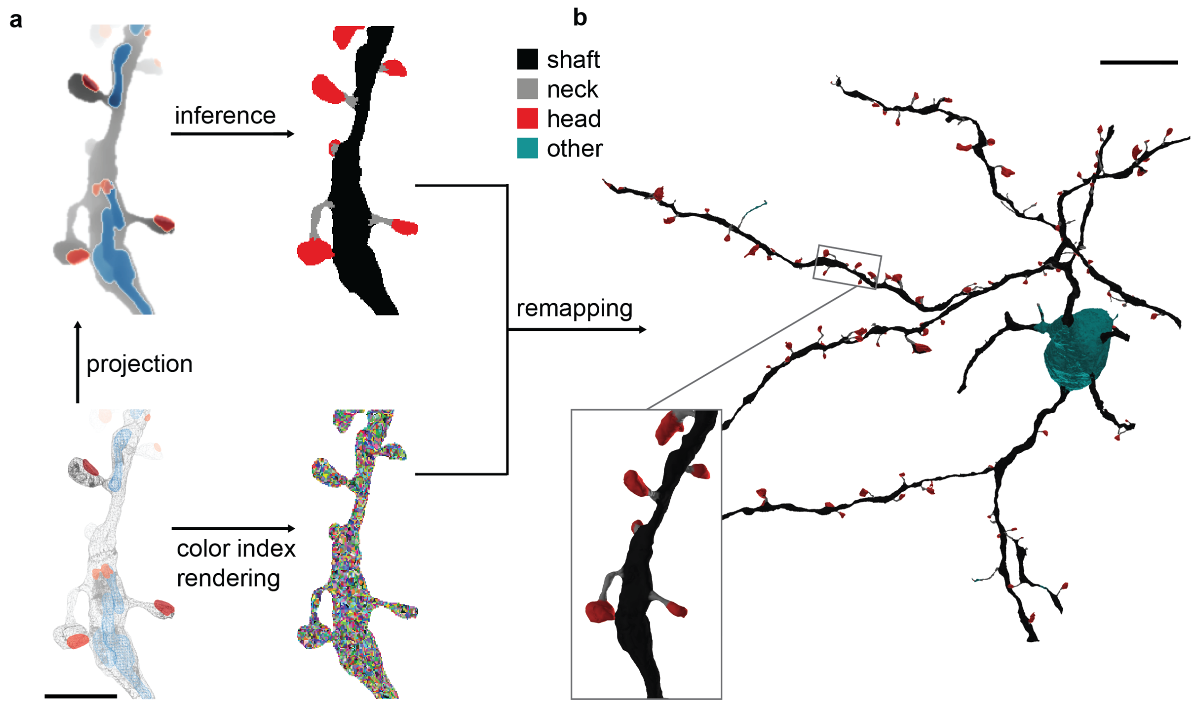


Figure 4.4: High-resolution surface segmentation of cell reconstructions. **a** The meshes of cell and assigned ultrastructure (mitochondria in blue, synaptic junctions in red) are rendered as orthographic projections into a morphology view (top left, depth and type encoding) and an index view (bottom right, only cell mesh). The inference results in pixel-wise labels that are combined with a vertex lookup stored in the index view. **b** Thus, the sparse view predictions, each one capturing only a fragment of the cell, are remapped and combined to form a semantic segmentation of the neuron surface. Figure adapted from (Schubert et al. 2019).

coded in RGBA space (index view). Since the same camera perspective is used for this second rendering pass, the semantic segmentation results can be readily associated with their original mesh vertices. Using 8-bit unsigned integer precision per RGBA channel, an one-to-one mapping between RGBA space and scalar vertex or face index allows the processing of meshes with up to $256^4 - 1$ vertices, with one ID being reserved to encode background.

The semantic segmentation is carried out on the morphology view M with a 2D CNN, that learned a transformation to pixel-wise class probabilities outputting the label view $L = f(M)$. Fig. 4.4a shows the CNN output with 5 classes, including dendritic shaft (black), spine neck (gray), spine head (red), one class for axon/soma (turquoise) and background (white).

This approach does not guarantee that a class label is assigned to all mesh vertices (due to visibility), which is why classifications are propagated to the mesh vertices by assigning pixel labels $L_{x,y}$ from the label view to the vertices associated with the IDs stored in the corresponding index-view pixels $I_{x,y}$ for all locations (x, y), $x \in [1, \ldots, N_{pix,x}]$ and $y \in [1, \ldots, N_{pix,y}]$. Note that this can be done either by storing face IDs (vertex indices forming a triangle), or the vertex IDs directly. In the case of face IDs and triangle meshes, one pixel label consequently propagates to three vertices. The index view example in Fig. 4.4a contains colors associated to face indices and used $N_{pix,x} = 256$ and $N_{pix,y} = 128$. Finally, the classification of a vertex is performed by finding the majority class in all the labels associated to it (Fig. 4.4b).

The identification of the three major compartments dendrite, soma, and axon in (Schubert et al. 2019) was realized by classifying multi-views in image-to-scalar fashion, which suffers from a low resolution and requires the rendering of multi-views for every prediction location. Building on the proposed semantic segmentation approach of surfaces with multi-views, new ground truth was generated to learn models for the high-resolution semantic segmentation of dendrites (shaft, spine neck, spine head and axon/soma) and a coarse semantic segmentation into the three major neuron compartments and finer axon structures (boutons).

Table 4.8: Performance values of the multi-view model on the segmentation task of dendrites (*zebra finch area X, small*) depending on the number of used views and the number of nearest neighbors denoted in brackets as (views, nearest neighbor).

	(1 view, 1 n.n.)	(1, 20)	(2, 1)	(2, 20)	(6, 20)
neck	0.703	0.727	0.714	0.731	0.746
head	0.865	0.874	0.878	0.880	0.886
shaft	0.961	0.968	0.958	0.966	0.973
average	0.843	0.856	0.850	0.859	0.868
accuracy	0.907	0.918	0.910	0.918	0.926

The model for the segmentation task of dendrites was trained on five neuron reconstructions (*semseg-fine-train*, Appendix B), with the main goal of separating spine head from dendritic shaft. The F_1-scores of dendritic shaft and spine head on the vertices of a dendritic branch (*semseg-fine-test-vertices*, Appendix B) were only slightly affected by the number of generated views (Table 4.8) and a higher level of smoothing with a k-nearest neighbors (k-NN) classifier (Fix and Hodges 1989). For most of the experiments the setting with 2 views a 20-NN classifiers was applied, due to the saturating gain of more views.

Next, the coverage of predictions was measured dependent on the number of views per rendering location (more views decrease the rotation angle between two subsequent projections in a multi-view). All unique face IDs that were part of a cell's index views were counted and then divided by the total face count to obtain the prediction coverage. The coverage reached approx. 70% with 4 views and converged to 81% with 15 views (Fig. 4.5), ensuring a good coverage already with 2 views (60%). The level-off effect below 1.0 is explained by the meshing procedure, which is performed on supervoxel level. Hence, agglomerated supervoxels form surfaces that are completely occluded inside the cell at the

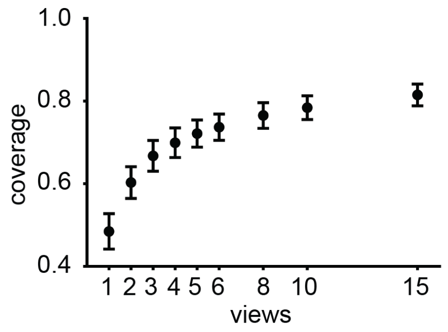


Figure 4.5: Fraction of processed neuron surface. The plot shows the mean fraction and 1-σ interval of predicted mesh faces with respect to the number of projections per multi-view (2 µm voxelization) calculated on five cell reconstructions (*dendritic-synapses*, Appendix B). Figure adapted from (Schubert et al. 2019).

merge location.

In addition to the vertex-level performance, the prediction was evaluated on a set of manually annotated synapses (*dendritic-synapses*, Appendix B) to quantify the effective performance in the synaptic connectivity matrix. This resulted in a much higher F_1-score of 0.978 (precision 0.978, recall 0.978, F_1-score spine head only 0.977; 2 views per location, $k = 20$). Synapse predictions were found by majority vote of the 20 nearest vertex labels relative to its center coordinate.

Although a single bouton class would suffice in general, the terminal bouton class was introduced in addition to the bouton *en-passant*, to probe the model sensitivity. Single-class bouton predictions can be combined with information of the skeleton graph to identify end-points which in turn allows subsequent separation of terminal and en-passant type, rendering the terminal bouton optional.

The model for the coarse segmentation (*semseg-coarse-train*, Appendix B) achieved an unweighted class average F_1-score of 0.796 (dendrite: 0.986, axon: 0.877, soma: 0.992, bouton: 0.776, terminal: 0.350; accuracy: 0.953) evaluated on the skeleton nodes of 6 reconstructions (*semseg-coarse-test*, Appendix B) and a similar score on the vertices with 0.816 average F_1-score (accuracy: 0.935). Unpredicted vertices were labeled using a nearest-neighbor classification ($k = 20$) and skeleton nodes with $k = 50$. The k-NN majority vote on skeleton node level slightly improved the accuracy (average F_1-score 0.806 and accuracy 0.926 with $k = 1$), while the main source of errors originated from the two bouton classes. Reducing the two bouton classes to a single one yielded a high F_1-score of 0.945 on skeleton nodes (dendrite: 0.986, axon: 0.877, soma: 0.992, bouton: 0.923; accuracy: 0.973).

The low performance of the terminal bouton class could presumably have been caused by a combination of lower class support with respect to bouton *en-passant* (4x on the training set) and boundary artifacts, where boutons *en-passant* might appear as terminals. Importantly, the relevant overall bouton performance is considerably higher, which allows to subsequently separate terminal and *en-passant* using the cell skeleton. The two classes were combined for all other experiments.

Next, the models trained on the *zebra finch area X, small* were applied on the approximately 10-fold larger *zebra finch area X, large* data set. The segmentation of the coarse and fine model was combined by extending the coarse dendrite prediction with the fine spine predictions (dendritic shaft, spine neck, spine head). Despite different EM image voxel sizes ($9 \times 9 \times 20\,\mathrm{nm}^3$ vs. $10 \times 10 \times 25\,\mathrm{nm}^3$), the mesh reconstructions reside in isotropic world coordinates (i.e. scaled to nanometers) and the transition of the models is straightforward. To assess the transferability of the coarse and fine model quantitatively, the prediction results were evaluated on a new ground truth data set consisting of 13 annotated neurons (*semseg-large-test*, Appendix B). Manual annotation was sped up by using sparse labels and including visualizations of the surface predictions of the previous models to guide human annotators (Materials and Methods).

Table 4.9: Performance values of the multi-view approach on *semseg-large-test* (Appendix B). The evaluation was performed on vertices and nodes of all 6 classes (C_6; dendrite, axon, soma, bouton, terminal, neck, head) and the major three compartments (C_3; dendrite, axon, soma).

	C_6 (vertices)	C_6 (nodes)	C_3 (vertices)	C_3 (nodes)
dendrite	0.767	0.754	0.864	0.885
axon	0.807	0.878	0.951	0.964
soma	0.929	0.920	0.929	0.920
bouton	0.772	0.683	-	-
neck	0.494	0.550	-	-
head	0.332	0.461	-	-
average	0.683	0.708	0.915	0.923
accuracy	0.791	0.801	0.924	0.943

Likely due to the larger data diversity, now including more cell types than in the small data set, the evaluation yielded an average node F_1-score of 0.680 and accuracy of 0.793 (Table 4.9). Slight changes in the meshing resulting from the different voxel sizes in the data sets may also have contributed. Reducing the fine predictions to the major three compartments (bouton to axon, head and neck to dendrite) resulted in a high performance for both vertices (avg.: 0.915, acc.: 0.924) and nodes (avg.: 0.923, acc.: 0.943), reflecting the challenging segmentation of fine structures, in particular spine heads. Nonetheless, this shows that models operating on neuron meshes transfer easily with decent prediction performance, allowing their labels to be used for initial inspection of unseen VEM data sets.

Point model comparison

In order to process the larger EM volume with models that take a more diverse set of neuron types into account, the ground truth that was used to evaluate the multi-view prediction was further extended by a large training set (*semseg-large-train*, Appendix B). The multi-view approach requires to cache the input and target projections on disk because the overhead of on-the-fly rendering is too heavy during training. Due to training set size with over 100,000 labeled nodes this would result in more than 1 TB[6] of required disk storage.

Table 4.10: Performance values of the point models on (*semseg-large-test*, Appendix B). The evaluation was performed on vertices and nodes of all 6 classes (C_6; dendrite, axon, soma, bouton, terminal, neck, head) and the major compartments (C_3; dendrite, axon, soma).

	C_6 (vertices)	C_6 (nodes)	C_3 (vertices)	C_3 (nodes)
dendrite	0.848	0.831	0.879	0.908
axon	0.809	0.879	0.959	0.971
soma	0.922	0.905	0.922	0.905
bouton	0.761	0.672	-	-
neck	0.513	0.498	-	-
head	0.624	0.615	-	-
average	0.746	0.733	0.920	0.928
accuracy	0.820	0.818	0.931	0.953

As a less pre-processing intensive alternative, continuous convolutions (Boulch 2020) were adopted to directly process point clouds. For this purpose the semantic segmentation of neuron surfaces was divided in multiple semantic groups (axon, dendrite, soma; dendritic shaft, spine neck, spine head; axon, bouton *en-passant*, terminal bouton), resulting in a hierarchy of three point morphology networks. One model was specialized for each group (input parameters: r_{ctx} of 15 µm and 15,000 points; Materials and Methods) and trained on *semseg-large-train*. Like before, vertex predictions were smoothed with $k = 20$ and mapped to skeleton nodes with $k = 50$ neighbor majority.

Not surprisingly, the average F$_1$-score of all 6 classes (dendrite, axon, soma, bouton, neck, head) increased by about 6% on vertex and 3% on node level compared to the transfered multi-view models, with the largest improvement on the fine spine head structures (0.624 vs. 0.332 on vertex level). The average performance difference of axon, soma and dendrite was only marginal (0.928 vs. 0.923), again reflecting the good transition of the approach to the second EM volume. Remaining errors occurred predominantly close to the soma, where in some cases it was ambiguous where compartment started, or parts with

[6] Assuming 100,000 locations, 4 projections per location, 1024 × 512 8 bit pixels per projection with $4 + 1$ input and target channels, the total size results in approx. $10^5 \cdot 4 \cdot 0.5 \cdot 10^6 \cdot 5\,\mathrm{B} = 1\,\mathrm{TB}$ only for the coarse segmentation model.

Table 4.11: Performance values of the point models without ultrastructure information on (*semseg-large-test*, Appendix B). The evaluation was performed on vertices and nodes of all 6 classes (C_6; dendrite, axon, soma, bouton, terminal, neck, head) and the major compartments (C_3; dendrite, axon, soma).

	C_6 (vertices)	C_6 (nodes)	C_3 (vertices)	C_3 (nodes)
dendrite	0.257	0.281	0.300	0.383
axon	0.675	0.800	0.841	0.884
soma	0.867	0.852	0.867	0.852
bouton	0.606	0.515	-	-
neck	0.303	0.299	-	-
head	0.433	0.427	-	-
average	0.523	0.529	0.669	0.706
accuracy	0.615	0.671	0.749	0.804

fewer defining ultrastructural features, like the initial segments of axon and dendrite. As expected, without any ultrastructure information, i.e. only using the point cloud of the cell surface during training and inference, the overall performance dropped drastically to 0.529 on the node level (Table 4.11).

Table 4.12: Performances of the point models on the test GP and MSN neuron and transfered multi-view (m.v.) performance on MSN.

	GP	MSN	MSN (m.v.)
dendrite	0.877	0.936	0.939
axon	0.854	0.881	0.866
soma	0.976	0.973	0.982
bouton	0.815	0.685	0.561
neck	0.474	0.699	0.759
head	0.037	0.805	0.825
average	0.672	0.830	0.822
accuracy	0.873	0.888	0.892

A closer look at the performance of the most frequent cell type in area X (medium spiny neuron, MSN) revealed a vertex F_1-score that was much higher (0.830, Table 4.12) than the overall average (0.746, Table 4.10), with a similar spine head score as the multi-view approach on the small EM volume (Section 4.3.1), which lead to a high synapse-level precision. The same was true for the performance values of the transferred multi-view models (0.822 vs. 0.683). Spines in pallidal-like cells (GP) in contrast are very infrequent (see Fig. 4.6a) and were barely identified correctly (0.037 on vertex and 0.108 on node level), which might be a result of the reduced data availability on the test set (31 annotated head nodes in GP vs. 1059 in MSN[7]) and very rare occurrences in the training set.

Examples of remaining errors are presented for the test GP and MSN neurons, which show that especially the boundary regions between two classes (e.g. between axon and its boutons, or between spine neck and head/shaft) are affected (Fig. 4.6). These transition regions, similar to the initial segments of axon and dendrite, are hard to label definitively as annotations are done on node level and can even be ambiguous (see also protrusion in Fig. 4.6b). Overall, the predictions result in a morphologically consistent partition of the surface.

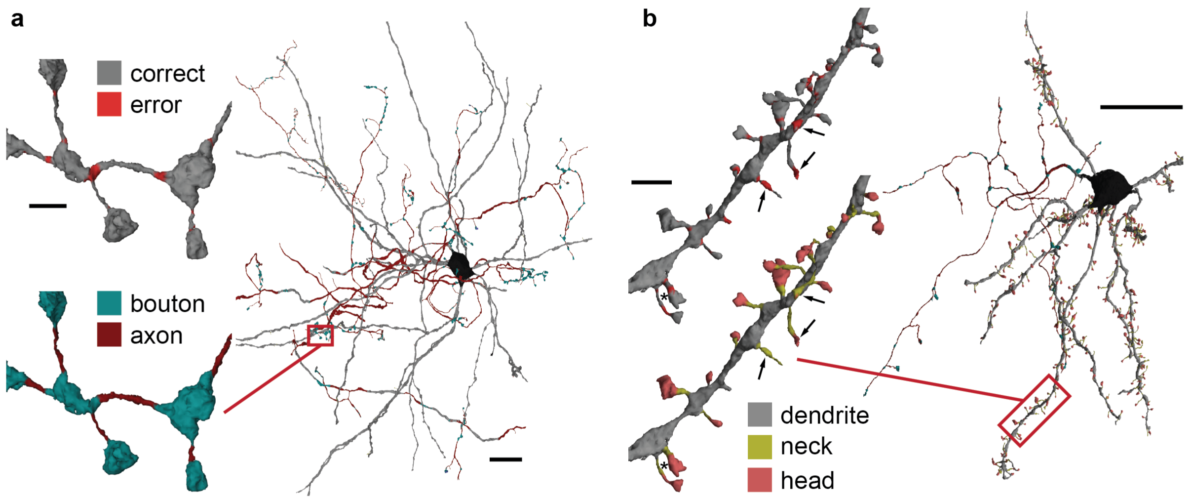


Figure 4.6: Example point-model predictions of a GPi (**a**) and an MSN (**b**) test cell (the same as in Table 4.12). Each panel shows on the left erroneous vertices (in red) and the corresponding prediction inset. Prediction colors correspond to dendrite (gray), axon (dark red), soma (black), spine neck (yellow), spine head (bright red). The arrows in **b** indicate errors at protrusions that are also manually hard to classify, and the asterisk typical errors at neck-head boundaries. Scale bars are 20 µm for the whole cell renderings and 2 µm for the insets.

Next, the point model performance was assessed on the small EM volume by performing an evaluation on the vertices of the *semseg-coarse-test* ground truth (dendrite: 0.982, axon: 0.809, soma: 0.971, bouton: 0.820; avg.: 0.895; accuracy: 0.933) and on skeleton nodes (0.979, 0.814, 0.965, 0.803; avg.: 0.890; acc.: 0.946) using the same input properties as previously with a context radius r_{ctx} of 15 µm and 15,000 inputs points. A comparison to the multi-view models showed a slightly degraded performance on node-level with a class-average F_1-score of 0.890 (mutli-views: 0.945) and accuracy of 0.946 (0.973).

To further evaluate the point approach on synapse level (*dendritic-synapses*), two point models were trained for the dendrite segmentation task. One for coarse (dendrite and non-dendrite) and one for fine segmentation (shaft, neck, head). Due to the comparably small

[7]A single spine head may consist of multiple nodes.

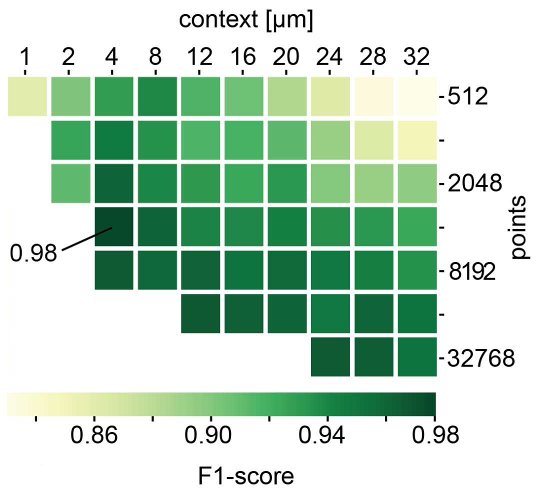


Figure 4.7: Grid search of context radius and number points for the point-based compartment predictions, evaluated on a set of manually labeled synapses (*dendritic-synapses*, Appendix B). Figure adapted from (Schubert et al. In review).

training data set, leading to quick trainings, the influence of the input parameters was assessed by conducting a grid search of the number of input points and the context radius r_{ctx} (Materials and Methods), where the coarse model was fixed and the input to the fine model was varied. For every parameter pair, three trainings were run, resulting in high mean F_1-scores over a wide value range, preferably for balanced context-to-point ratios (4 µm, 4k points: 0.969, 0.980, 0.986; 16 µm, 16k points: 0.963, 0.952, 0.980; Fig. 4.7).

4.3.2 Cell type classification

In (Schubert et al. 2019), high-accuracy results for the prediction of four cell type classes (excitatory axon, medium spiny neuron, interneuron, pallidal-like neuron) of the *zebra finch area X, small* data set were presented. Due to the larger volume of *zebra finch area X, large*, it contains more complete neuron reconstructions and consequently allowed a finer, manual morphological separation into 11 putative types (*celltypes-large*, Appendix B).

For this task, the input point features were extended by including axon myelination[8] and synapse type (excitatory or inhibitory, see Chapter 5). As a result, the dimensionality of the one-hot encoding of the input point features increased from previously 4 (cell surface, synaptic junctions, mitchondria, vesicle clouds) to 6 dimensions (cell, myelinated cell, excitatory syn., inhibitory syn., mito., v.c.).

Table 4.13: 10-fold cross-validation F_1-scores and accuracy of the cell type classification using a context radius of $r_{ctx} = 20\,\mu m$, 50,000 input points and three repetitions for redundancies 50 and 1.

	redundancy $N = 50$			redundancy $N = 1$		
repetition	0	1	2	0	1	2
STN	0.945	0.931	0.944	0.8	0.684	0.732
DA	1	0.972	1	0.944	0.812	0.944
MSN	1	1	1	0.923	0.941	0.939
LMAN	1	1	0.984	0.906	0.843	0.895
HVC	1	1	1	0.857	0.918	0.903
TAN	1	1	1	0.857	0.782	0.695
GPe	0.866	0.892	0.857	0.687	0.645	0.592
GPi	0.875	0.941	0.882	0.758	0.838	0.750
FS	0.961	0.961	0.961	0.692	0.692	0.716
LTS	0.888	0.842	0.842	0.640	0.555	0.545
NGF	1	0.978	1	0.857	0.830	0.893
average	0.958	0.959	0.951	0.811	0.776	0.782
accuracy	0.968	0.968	0.964	0.826	0.798	0.814

The model performance was evaluated by 10-fold cross-validation with a redundancy of three training repetitions per split (training: 90%, test: 10%; 30 models in total, each trained for about 7 days). The unweighted class-average F_1-score reached 0.951 in the worst and 0.959 in the best case, with GPe, GPi and LTS being the hardest types (Table 4.13).

Fig. 4.8 shows the result of a parameter search for context radius, number of input points and the number of predictions per cell (redundancy) with the overall mean F_1-score for varying input parameters. If the model input was represented by too few points (20 µm,

[8]The myelination information was aggregated on skeleton node level from a dense voxel prediction and then propagated to the associated cell surface vertices. See also Chapter 5.

5k), which translates into a low level of detail, or the input did not contain sufficient context (4 µm, 25k), the performance of individual predictions was overall poor. For a fixed context of 20 µm, performance increased substantially up to a number of 75,000 points (mean F_1-scores with $N = 50$ for 25k: 0.954, 50k: 0.956, 75K: 0.944).

To test the impact of redundancy on the predictions for averaging, the model was applied to a range of $N \in [1, 10, 20, 50]$ inputs, generated at random locations of the cell. The individual class predictions for $N > 1$ were combined to a single prediction by majority vote of the individual hard classifications. The performance gain is very prominent between $N = 1$ and $N = 10$ (Table 4.13) and is less apparent for larger N and large contexts (Fig. 4.8).

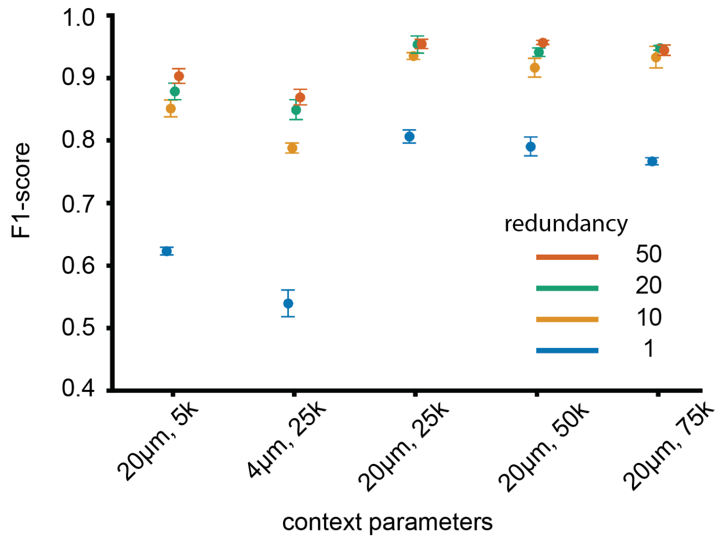


Figure 4.8: Classification performance of putative cell types dependent on the input parameters (context radius, input points) and the number of generated inputs (redundancy) per neuron. E.g. 20 µm, 5k refers to a 20 µm context radius with 5,000 points. The data points are the overall mean ± s.d. of the unweighted class-average F_1-score of three repetitions. One repetition of an input parameter configuration consisted of 10 trainings (10-fold cross-validation).

The baseline model (50,000 points, 20 µm context) with a redundancy of $N = 50$ achieved an overall mean F_1-score of 0.956 (averaged across the three repetitions) on the 10-fold cross-validated ground truth, compared to 0.930 F_1-score with the multi-view approach (Table 4.14, Materials and Methods). The major performance drop originated from the LTS class with a mean F_1-score of 0.692 and STN with 0.878.

The impact of missing myelin and synapse type information on the model performance was assessed by conducting a feature ablation study where models were trained on the same samples, but with a reduced input feature set. Excluding myelin reduced the overall mean F_1-score by 0.023 and had a severe effect on the performance of LTS (mean F_1-score 0.730 vs. 0.857). This performance drop is similarly visible in the results of the multi-view approach, which did not incorporate myelin information in the 2D projections (Table 4.14). The ablation experiment of the synapse type in contrast did not show a performance degradation with a mean F_1-score of 0.958.

Table 4.14: 10-fold cross-validation F_1-scores and accuracy of the cell type classification without myelin and using multi-views. Other than the feature dimension, the input parameters for the point model are the same as in Table 4.13 with $r_{ctx} = 20\,\mu\text{m}$, 50,000 input points and a redundancy of 50. The multi-view models used $N = 20$ projections with $l_w = 8\,\mu\text{m}$ and $l_h = 4\,\mu\text{m}$ (Section 4.2).

	w/o myelin			multi-views		
repetition	0	1	2	0	1	2
STN	0.888	0.916	0.906	0.857	0.895	0.882
DA	0.972	0.972	1	0.950	1	0.947
MSN	0.969	0.969	1	1	1	1
LMAN	1	0.967	1	0.967	0.949	0.966
HVC	1	0.984	1	0.952	0.985	0.939
TAN	0.956	1	1	1	1	1
GPe	0.857	0.896	0.896	0.933	0.896	0.896
GPi	0.882	0.909	0.909	0.937	0.909	0.909
FS	0.872	0.925	0.961	0.941	0.961	0.961
LTS	0.750	0.736	0.705	0.645	0.740	0.692
NGF	0.978	0.977	1	1	1	1
average	0.920	0.932	0.943	0.925	0.939	0.926
accuracy	0.936	0.944	0.960	0.932	0.948	0.936

Further, the mean of the N individual class probabilities $p_i = \frac{1}{N}\sum_k^N p_{i,k}$ was calculated to estimate the model uncertainty based on the information entropy of the predictions:

$$\begin{aligned}\text{certainty}(\mathbf{p}) &= 1 - H(\mathbf{p})/H_{max} = 1 + \frac{1}{H_{max}}\sum_{i=1}^{C} p_i \log_2 p_i \\ &= 1 + \sum_{i=1}^{C} p_i \log_C p_i\end{aligned} \tag{4.5}$$

The maximum entropy is given by

$$H_{max} = -\sum_{i=1}^{C} \frac{1}{C}\log_2 \frac{1}{C} = \log_2 C \tag{4.6}$$

The certainty of the point model with $r_{ctx} = 20\,\mu\text{m}$ and 50,000 points on the test data (for all three repetitions), split into correct and incorrect classification is shown in Fig. 4.9 and differs significantly (two-sided Mann-Whitney U test (Mann and Whitney 1947) statistic: 5.24 and p-value: $1.57 \cdot 10^{-7}$; correct: 734, incorrect: 25), which makes this a useful quantity for guided proofreading of cell type ground truth data.

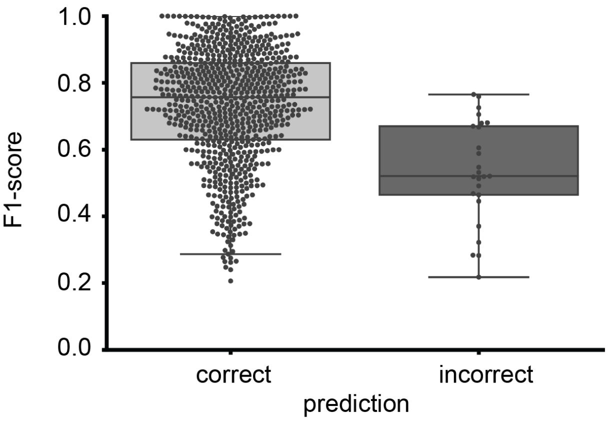


Figure 4.9: Model certainty on the 10-fold cross-validated cell type classification ground truth. Box plot represents the median, lower and upper quartile; whiskers are 1.5x interquartile range (Q3-Q1). Points are the values of all three repetitions using 20 µm context radius, 50,000 points and a redundancy of 50.

4.3.3 Cell clustering

With the availability of ever larger data sets in VEM, it becomes possible to identify patterns, which can reduce manual annotation efforts (Chen et al. 2020) or provide an assessment of the data with little human bias. Unsupervised or self-supervised approaches don not require explicit target labels and aim to extract feature representations that lead to semantic arrangement of the input data, which in turn can be used to improve subsequent supervised tasks. Self-supervised metric learning, as ,for example, the *SimCLR* framework presented in (Chen et al. 2020), alters one sample from the original distribution by augmentation to generate a pair, of which both instances are semantically similar but inherit substantially different input statistics. During training, the model learns to push these artificially generated pairs together in a latent space and different samples from the original distribution apart from each other.

For the purpose of learning cell morphology embeddings, the previous self-supervised approach using multi-views (Schubert et al. 2019) with triplet loss (Schroff et al. 2015) is here extended for embedding entire neurons directly from local point clouds. A model was trained to extract a 10-dimensional latent feature vector from an input cloud, with the constraint to keep “similar” inputs closer than “dissimilar” in a learned latent space. Two point cloud contexts ($r_{ctx} = 15$ µm, 25k points) within 15 µm distance in the same cell were treated as similar, and a third context, drawn randomly from any other cell in the *zebra finch area X, large* volume as dissimilar (Materials and Methods). The model was applied to

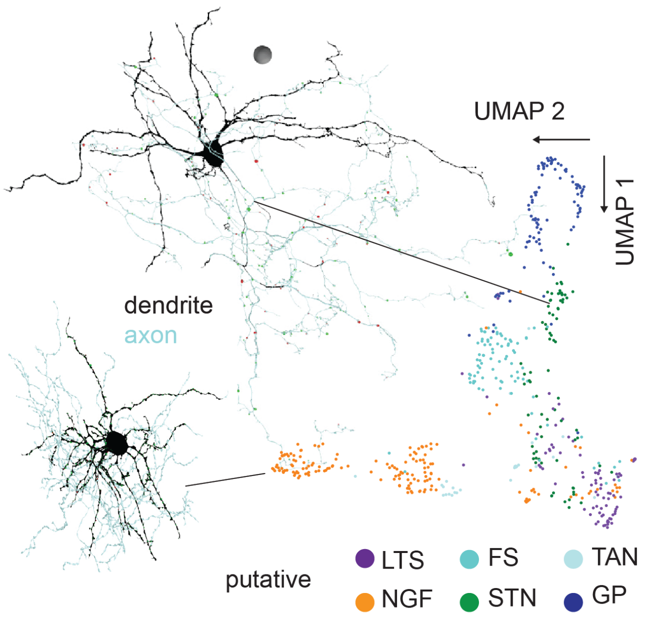


Figure 4.10: Self-supervised neuron clustering of the latent space of 531 neurons in the data set that contained soma, axon and dendrite (MSNs not considered). The embedding vectors were transformed using a UMAP dimensionality reduction. Colors indicate putative cell type based on supervised classification with the point model. Scale sphere has a 10 µm diameter. Figure adapted from (Schubert et al. In review).

the whole *zebra finch area X, large* data set and the inspection of the resulting embeddings was focused on the rare cell types of area X, only considering cell reconstructions with a soma skeleton length > 10 µm (compartment predictions were smoothed using a majority vote on all node labels collected within 10 µm path length), axon and dendrite skeleton lengths > 200 µm, and that were not classified as MSN or an axon class only projecting to area X (LMAN, HVC, DA). For each of the resulting 531 cells a 10-dimensional compound latent vector was constructed by averaging the local triplet-loss embeddings along each dimension separately for axonal and dendritic compartments, followed by summation of the two vectors.

The low-dimensional 2D UMAP (McInnes et al. 2020) projection[9] of the compound latent space, colored by the prediction results of the supervised classification with the point model from Section 4.3.2, indeed formed clusters of known morphological neuron types (Fig. 4.10), such as putative cholinergic (TAN) and pallidal-like neurons (GP). Furthermore, the UMAP projection suggests that area X contains more cell types that can morphologically be distinguished, for example a type of local neurons that form synapses with excitatory ultrastructural characteristics (STN), that has so far only been physiolog-

[9]Parameters: n_neighbors=60, metric='euclidean', random_state=0, min_dist=0.05, n_epochs=1000

ically identified (Budzillo et al. 2017) but not anatomically characterized. This provides a first glimpse into the expressiveness of dense morphology information contained in connectomic EM data, which could become a powerful tool for the characterization of neuron types in a brain area.

4.4 Discussion

Recently, many sophisticated approaches for instance segmentation of neurons (Januszewski et al. 2018; K. Lee et al. 2021; Sheridan et al. 2021) and pipelines for acquiring and processing large EM volumes (Shapson-Coe et al. 2021; Yin et al. 2020) have been introduced. One has to now analyze all this data, and processing the surface of neurons and ultrastructure offers a promising alternative to voxel inputs. In this thesis, two approaches for compartment identification were compared, that are based on semantic surface segmentation either using multi-views or point clouds. The proposed methods are more data-efficient in terms of required throughput than previously published approaches based on voxel-volume (H. Li et al. 2020) or multi-view classification (Schubert et al. 2019), and were tested down to a resolution of individual spine necks and heads. In addition, methods that use dense voxel grids suffer from an increasing fraction of background with larger field of views (H. Li et al. 2020).

The evaluations on two EM data sets showed a similar high-level performance for both point and multi-view networks in predicting the major three neuron compartments and in predicting head vs. shaft at the synapse level, with a slight advantage on the computationally more intensive multi-view approach. The surface segmentation of all cell types in area X remains challenging, in particular of rare structures, such as spines in pallidal-like cells.

Surprisingly, the multi-view models transferred to the large EM data set performed as well as the point models on medium spiny neurons, the most frequent cell type in area X, even though the point models were trained on data-set-specific ground truth. This reflects an input property that is generally desirable for connectomics, namely that the model input is independent of the voxel size and intensity characteristics of the underlying EM volume, at least in a narrow range. Given a minimum level of detail that can be represented by the resulting surface reconstruction of the neurons[10], transfer of the model to new data sets is straightforward as long as the data diversity is sufficiently covered. This facilitates the development of a general backbone model that is shared across different laboratories.

For the overall excellent classification performance of eleven putative classes, the inclusion of axon myelinization proved beneficial, whereas synapse type (inhibitory vs. excitatory) had no impact. The information entropy calculated from the predicted class probabilities differed significantly between correct and incorrect predictions, which could be exploited for guided proofreading and presumably the identification of out-of-distribution samples, such as neuron segmentations with a merge error combining two different cell types.

[10]The finest voxel size used for the point model input was 70 nm and the 30 nm pixel size for the multi-views.

A point morphology network was trained using self-supervised metric learning to embed point clouds of neuron fragments which were then aggregated to compound feature vectors on neuron level, using the (supervised) compartment labels. The clusters in the resulting neuron embeddings matched qualitatively well with class labels predicted by the supervised model, also for so far anatomically undescribed cell types in area X. In (Shapson-Coe et al. 2021), for example, the *SimCLR* framework (Chen et al. 2020) was adopted to separate two glial types with comparably little annotation effort. Self-supervision will likely become more important and powerful, in particular with increasing availability of unlabeled data.

The proposed dual representation of neurons using point clouds in combination with their cell skeleton could in future efforts also be ported to different, more recent architectures that evolved in the rapidly moving field for point cloud and mesh processing (Hanocka et al. 2019; Xiang et al. 2021).

Chapter 5

SyConn2 - A connectome analysis framework

This chapter contains text and figures from (Schubert et al. In review).

5.1 Introduction

With the detailed and dense reconstruction of neurons in VEM data sets, it is possible to study correlations of synaptic properties across different cell types with high statistical power. The high resolution enables the identification of structures such as, but not limited to, mitochondria, vesicle clouds, synaptic junctions, endoplasmic reticulum and Golgi apparatus. To date, these function-relating structures are detected on voxel level with CNNs (Buhmann et al. 2021; Dorkenwald et al. 2017; Haberl et al. 2018; Heinrich et al. 2021; Staffler et al. 2017) and can further be combined with the cell segmentation to build a database that consolidates circuit, neuron and ultrastructure properties.

The acquisition speed of VEM data sets has increased about 100-fold during the past 5 years (Kornfeld and Denk 2018), now yielding data sets of petabyte-scale (Shapson-Coe et al. 2021; Yin et al. 2020). The resulting computational challenges require scalable analysis solutions which can be run on classical high-performance computing (HPC) environments or cloud computing services and are facilitated by open-source code to increase reproducibility. Despite considerable advances in areas such as automated neuron reconstruction (Januszewski et al. 2018; K. Lee et al. 2017), proofreading (Dorkenwald et al. 2022; Zhao et al. 2018) and integrative processing in cloud environments (Johnson et al. 2020; Macrina et al. 2021), a pipeline that creates an annotated connectome and that can also be operated cost-efficiently on existing HPC infrastructure is lacking.

The synapse reconstruction framework published in (Dorkenwald, Schubert, et al. 2017) incorporated ultrastructure and membranes, and included cell analysis methods based on hand designed features. Since automated neuron tracing with reasonable run-lengths was not possible back then, it used manual cell tracings as starting point. The 3D cell membrane prediction was transformed into a sparse hull representation by a ray-casting approach that associated membrane sample points with the cell skeleton. To identify synaptic connections between a pair of neurons, contact sites, areas where the neuron hulls are in close proximity, were extracted and checked if they coincided with a predicted synaptic junction. In order to efficiently support the now available segmentation of the flood-filling networks, the previous framework *SyConn* (short for synaptic connectivity) was fundamentally upgraded.

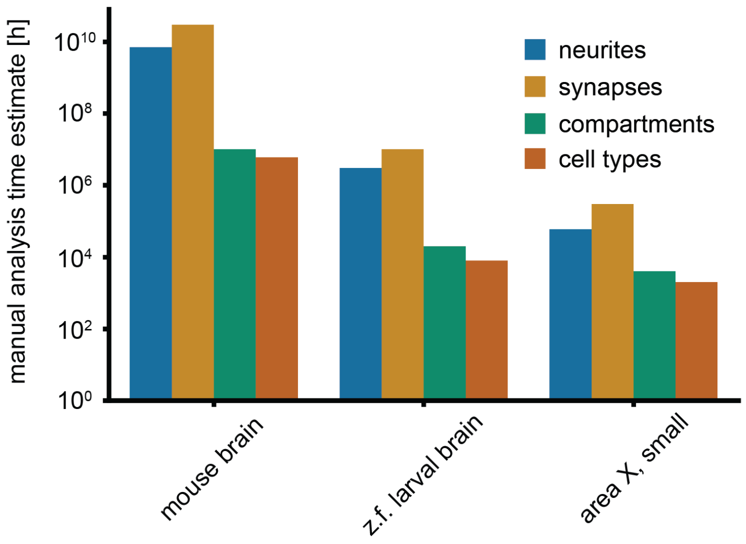


Figure 5.1: Time estimates of the different reconstruction steps for manual connectomic analysis of a whole mouse brain, zebra fish larval brain and *area X, small* (data taken from (Dorkenwald, Schubert, et al. 2017)). The time for neuron reconstruction refers to a volume reconstruction based on manual cell tracings and not a voxel-wise annotation.

The large number of neuron-neuron contacts in dense volumes of neural tissue, the fact that not every contact is synaptic (Kasthuri et al. 2015), and the central role of synapses make their careful reconstruction particularly important. Moreover, counter-intuitively, reconstructing synapses can take more time than neuron tracing when performed fully manually (Fig. 5.1).

This chapter outlines the design and implementation of the *SyConn2* analysis framework. SyConn2 combines morphology neural networks (Chapter 4) with detailed synapse extraction and allows neuroscientists to run queries against connectomes with millions of synapses.

5.2 Materials and Methods

Input segmentation maps and ultrastructure predictions

Zebra finch area X, large

The cell instance segmentation map was provided by M. Januszewski and generated using the flood-filling neural networks as reported earlier (Januszewski et al. 2018), with additional training data provided by annotators at the MPI of Neurobiology and ariadne.ai ag. Synaptic junction (sj), synapse type (symmetric and asymmetric), vesicle cloud (vc) and mitochondria (mi) voxel segmentation maps were equally provided by M. Januszewski using a 3D convolutional neural network model that predicts these classes on a per-voxel level, followed by thresholding.

A myelin segmentation map (4-fold downsampled) was generated using SyConn's neural network inference pipeline, that divides the data set into a configurable number of data chunks (used here: cube size of [482, 481, 236] voxels with additional [30, 31, 20] overlap on every side) to enable parallel processing. For the myelin inference, a model based on the U-Net architecture (Ronneberger et al. 2015) was trained using the elektronn3 framework (*myelin-gt*, Appendix B; class weights: 1, 2) with the following parameters: 32 output channels in the first layer, output channels increase by a factor of 2 in every downpath layer, 4 downpath and up-path layers, ReLU activation and batch normalization. Instead of an isotropic kernel size of 3, first and third layers in the down- and up-path had a z-kernel extent of 1 (planar block). For the training, ground truth generated on *zebra finch area X, small* was used and transformed by flip, grayscale, gamma, Gaussian noise and blurring augmentations.

Zebra finch area X, small

M. Januszewski provided the cell instance segmentation map generated with flood-filling neural networks. Voxel segmentation maps for sj, vc and mi were predicted using CNNs (Dorkenwald, Schubert, et al. 2017). The myelin prediction was performed with a U-Net architecture similar as above, but with 16 output channels in the first layer. The synapse type (*synapse-type-gt*, Appendix B) was equally predicted via a U-Net with a total of 4 blocks, but without planar blocks and 28 initial channels and three final outputs. The manual ground truth annotations (background, symmetric, asymmetric) were extended by small cubes containing sj predictions with known synapse type, based on manually annotated pre- and postsynaptic cell type labels. Background voxels were added by 3-fold binary dilation of the foreground synapse type voxels. Remaining voxels were assigned an auxiliary ignore label. Training patches with a background ratio of more than 90% were skipped and the same augmentation scheme used for the large data set was applied.

Compute infrastructure

The development and implementation was done on the MPCDF compute cluster described in Section 4.2.

SegmentationObject generation

This section describes the extraction steps that make single supervoxels and their properties in the cell and ultrastructure segmentations accessible for processing and analysis. The (binary) ultrastructure segmentation maps were transformed into an instance segmentation by a 3D watershed procedure (`segmentation.watershed` from skimage package; Van der Walt et al. 2014), that was performed on the distance transform (`filters.distanceTransform` from the vigranumpy package; Köthe 2000) of the input maps. The seeds for the watershed were generated from the morphologically modified (vc: binary opening, binary closing, binary erosion; mi: binary opening, binary closing, 3x binary erosion) input maps using connected component analysis (`ndimage.label` from the scipy package; Virtanen et al. 2020). Compute tasks were distributed across the workers by chunking (512 voxels edge length; 6, 2 voxels overlap for mi, vc). Chunk-wise IDs were made unique data set wide, and the overlap regions were used to unify IDs of objects that span across multiple chunks. The resulting 3D connected components of voxels (supervoxels) were subsequently analyzed and stored in an accessible format, as described in the next paragraph.

The supervoxels formed the basis for SegmentationObjects (SO), which store additional properties (representative coordinate, voxel bounding box, voxel count, mesh, skeletons, mesh area and mesh bounding box) of cells, ultrastructure (mi, vc), contact sites (cs, see Synapse-cell association) and synapse fragments/agglomerates (sv-syn, see Synapse-cell association) and are collected in SegmentationDatasets (SD), with separate SDs for each SO type. An SD can therefore be seen as a key-value store that provides an interface to individual supervoxels/SOs.

The SO property extraction was performed on 3D chunks (512 voxels edge length) of every ultrastructure's instance segmentation, as described in the following: The mesh, voxel count, bounding box and representative coordinate of all segmentation IDs in a cube are computed in a single pass and the partial results are merged in a final reduction step. Representative coordinates were chosen as a random voxel coordinate (except for putative synapse objects, see Synapse-cell association), as long as it was inside the object. For every *syn* object the fraction of overlapping symmetric and asymmetric voxels was determined. Cell SO also store the ID and fraction of overlapping ultrastructure segmentation voxels and were skeletonized[1] using kimimaro (Silversmith et al. 2021) which adapts and extends the TEASAR algorithm (Sato et al. 2000). Meshes of cells, mitochondria and vesicle clouds were computed with zmesh[2].

[1]`scale=2` and `const=500`

[2]https://github.com/seung-lab/zmesh, meshing parameters were `simplification_factor`: 50, `max_simplification_error`: 40 nm.

SuperSegmentationObject generation

In this section, the extraction and generation of neuron properties is described, which provide access to neuron-level statistics, such as the size distribution of dendritic synapses. The SuperSegmentationObject (SSO) class represents the agglomerated supervoxels (cells or cell fragments) of a neuron segmentation. Based on a supervoxel graph, that defines which cell fragments belong to the same neuron, an SSO aggregates the properties of the corresponding cell SOs (representative coordinate, bounding box, mesh, skeleton) and contains associated ultrastructure SO IDs and further analysis results (cell type predictions and certainties, vertex and skeleton node compartment predictions, local morphology embeddings, spine head volumes, myelination status).

SO properties were merged as follows. Representative coordinate: first SO rep. coord; bounding box: min and max value of all SO bounding boxes; meshes: concatenation of vertices and indices; skeleton: concatenation of nodes and edges, adding edges between the closest skeleton nodes of SO skeleton pairs until the whole-cell skeleton was a single connected component.

Myelin predictions were mapped onto cell skeletons by storing the fraction of as-myelin-predicted voxels within a cube of size [11, 11, 5] voxels (voxel size [nm]: 40, 40, 100) at every skeleton node and thresholding (per-voxel probability threshold 0.5 and classification via majority vote). The myelin node predictions were smoothed using a running majority vote on all neighboring nodes collected within a 10 µm path traversal starting from the source node.

Vertex predictions of the morphology networks for dendrite, spines, axon, boutons and soma were propagated to skeleton nodes by calculating the majority vote of the $k = 50$ nearest prediction locations. The node labels were in turn smoothed using a running majority vote on all neighboring nodes collected within a 10 µm path traversal starting from the source node. The smoothing was only applied on the three major compartments (dendrite, axon, soma). Finer structures were then reassigned to corresponding compartments (e.g. bouton *en-passent* only if the original node remained axonic).

Synapse-cell association

The synapse reconstruction was performed through a multi-step extraction process. At first, a contact site instance segmentation was generated by iterating over the cell segmentation and storing adjacent supervoxel IDs. At every boundary voxel (6-connectivity) of the cell segmentation, a partner cell ID was identified by finding the majority ID within a window of [13, 13, 7] voxels (voxel size: $10 \times 10 \times 25\,\mathrm{nm}^3$). If a majority ID was found (background and the source boundary voxel ID were excluded) the contact site voxel was assigned a value that allowed the retrieval of the two partner cells (bit shift combination to uint64 in case of uint32 cell segmentation, tuple of uint64 in case of uint64 cell segmentation). The resulting thin boundary instance segmentation was morphologically closed (N=7 iterations; this is sufficient to close the maximum distance of adjacent cells found through the adjacency filter, see "contact sites" in Fig. 5.2) and dilated two times afterwards. Note that one instance in this segmentation represents all contact sites between a cell-supervoxel pair, since the contact instance ID is the same, even if the supervoxels touch at different locations.

In a second step, the contact site instances were intersected with the voxels of the *sj* foreground prediction, generating intermediate synapses only between cell supervoxels (sv-syn). Individual putative synapse objects between two cells (supervoxel agglomerates) were obtained by computing connected components on a graph that was built with the voxels of sv-syns of all the cells' supervoxels that form such sv-syns between the cell pair. Within sv-syns between the same supervoxel pair, edges were added between voxels not farther apart than two voxels, and sv-syns of different supervoxel pairs were connected if their closest voxels were within a distance of at most 250 nm. For generating synapse meshes the function `create_from_point_cloud_poisson` from open3D (Zhou et al. 2018) was applied on the voxels of the individual synapse objects. The area of the synaptic cleft was estimated by dividing the mesh area by two, to not double count the front - and back face. The representative coordinate of synapses was the voxel closest to the mean of all object voxel coordinates.

Synaptic properties

The resulting synaptic objects were further assigned a probability value with a random forest classifier[3] ($N = 10$ features: synapse size in voxels, mesh area, numbers and voxel counts of pre- and postsynaptic mitochondria and vesicle clouds), with 0 meaning least synaptic and 1 meaning most synaptic. For the training, a random set of extracted, putative synapse objects were manually annotated into synaptic and non-synaptic (*synapses-rfc*, Appendix B).

The voxel count features for nearby *mi* and *vc* objects (initial search within a maximal distance of 4 µm to the representative coordinate) were calculated by finding the number of *mi* or *vc* mesh vertices with a maximum distance of 2 µm or 1 µm to the synapse voxels respectively, followed by dividing this vertex count with the total object vertex count, to obtain a fraction that could then be multiplied with the object voxel count, resulting in the number used as feature (mesh vertices and synapse voxels were 2-fold subsampled).

The "spiness" of a synapse was identified with nearest-neighbor classification of the vertex labels predicted by the morphology network for dendrite segmentation ($k = 50$, relative to the synapse representative coordinate). An estimate for the spine head volume was calculated for all synapses that were labeled both as spine head from the vertices and dendrite from their nearest skeleton node (see SuperSegmentationObject generation). Centered at the synapse representative coordinate, all voxels of the binary postsynaptic neuron mask within a cube of $400 \times 400 \times 200$ voxels (original voxel resolution) were loaded. The neuron mask was resized to isotropic voxels, binary filled (`ndimage.binary_fill_holes` from scipy), and local maxima (`feature.peak_local_max` from skimage) of its distance transform (`ndimage.distance_transform_edt`) served as seeds for a watershed

[3]Using the scikit-learn (Pedregosa et al. 2011) implementation with 2000 trees.

(skimage) procedure. The watershed propagated the labels of the local maxima found by a 50-nearest-neighbor classification of the vertex spine predictions (spine neck, spine head, dendritic shaft). The spine head volume was then calculated from all the voxels of the segmented spine head supervoxel.

Time and cost analysis

The timing and throughput experiments were performed with a dynamically created SLURM cluster on Google Cloud Platform using elasticluster[4]. In total, 24 compute nodes (n1-highmem-32) each with 2 Tesla P100 GPUs, 32 virtual cores and 208 GB RAM were used in combination with a Gluster filesystem[5] (4 server nodes with SSD) and a 10 TB persistent disk to store the input data (aligned EM data, cell segmentation, myelin, sj, mi, vc and synapse type predictions). The processed data were subvolumes (voxel extent: [6144,6144,3072], [9216,9216,4608], [12288,12288,6144], [15360,15360,7680]) centered at the center of the *zebra finch area X, large* data set (Appendix A).

All processing steps were grouped as follows:

- Data store: Generation and caching of object properties for *mi* and *vc* SegmentationObjects (SO), and cell SO and SuperSegmentationObjects (SSO).
- Synapse extraction: SO generation (cs, sv-syns and putative synapses), feature extraction and RF classification.
- Synapse enrichment: Spine head volume estimation and consolidation of synapse properties.
- Morphological analysis (Chapter 4): Astrocyte prediction and splitting[6], cellular compartment prediction (Section 4.3.1, multi-view approach used the two models: axon, dendrite, soma, en-passant bouton, terminal bouton and dendritic shaft, spine neck, spine head, axon/soma; point-based used three models: axon, dendrite, soma and axon, en-passant bouton, terminal bouton and dendritic shaft, spine neck, spine head), cell type classification (Section 4.3.2; point model with 20 µm context, 50k points and redundancy of 20), morphology embeddings (Section 4.3.3, multi-view embeddings as described in Schubert et al. 2019).

The point models for the compartment prediction used the architecture as described in Table 4.4, with a slightly adapted configuration for the two fine-level models (neighborhood: [32, 32, 32, 16, 8, 8, 4, 8, 8, 8, 16, 16, 16], output points: [-1, 1024, 512, 256, 64, 16, 8, 16, 64, 256, 512, 1024, -1]) and the axon-dendrite-soma model (neighborhood: [16, 16, 16, 16, 8, 8, 4, 4, 4, 4, 8, 8, 8]).

[4]https://github.com/elasticluster/elasticluster

[5]https://www.gluster.org/

[6]Approach as described in (Schubert et al. 2019). For the astrocyte prediction with points, the point morphology learning networks (Section 4.1.2.) were slightly modified to infer skeleton node labels given an input point cloud context

Mito-synapse distance distributions

This analysis was performed on the *zebra finch area X, large* data set. The minimal distances between presynaptic MSN/GP (predicted GPi and GPe combined) synapses and mitochondria were calculated as the Euclidean distance between representative synapse coordinate and the closest mesh vertex (point on the surface; downsampled to voxels with edge length 200 nm) of the neuron's mitochondria. Neurons were filtered for minimal path lengths as follows to exclude small reconstructions: Minimum axon, dendrite and soma path length of 100 µm, 50 µm and 5 µm, respectively, and cell type certainty (Section 4.3.2) of at least 0.75. Only axo-dendritic synapses with a random forest classifier probability above 0.8 were included. Path lengths were calculated by summing the edges between cell skeleton nodes that were labeled as the respective compartment type. The inference of cell types was performed with the point model (Section 4.3.2), $N = 20$ predictions per neuron, 50,000 input points and a context radius of 20 µm.

The compartment prediction was performed with the multi-view morphology networks (Section 4.3.1) that were trained on the ground truth of the small data set (*area X zebra finch, small*; *semseg-coarse-train* and *semseg-coarse-fine*, Appendix B).

Cell types were predicted by the point model with 20 predictions per cell, 50,000 input points and a context radius of 20 µm (Section 4.3.2). During the assessment the information about synapse size (upper or lower half) and cell type was hidden.

A control for the minimal synapse-mitochondria distances was performed by sampling locations on the cell's axonal compartment surface randomly and calculating the distance to the closest mitochondria mesh vertex (downsampled to voxels with edge length 200 nm). For each cell up to 1000 skeleton nodes that belonged to the axon (less if the cell contained fewer nodes) were drawn, and for each node a random vertex from all cell mesh vertices that were assigned to that node via Voronoi partitioning was chosen as the control location.

5.3 Results

5.3.1 Circuit reconstruction

Taking advantage of the details visible in dense heavy metal staining of tissue, SyConn2 processing begins with multiple voxel-level semantic annotations that span the entire VEM data set, including segmentation into cells, extracellular space and ultrastructure (Fig. 5.2). It provides the option to apply deep neural network segmentation models to an entire EM data set by splitting it into chunks and distributing them to multiple workers using the SLURM workload manager (Yoo et al. 2003).

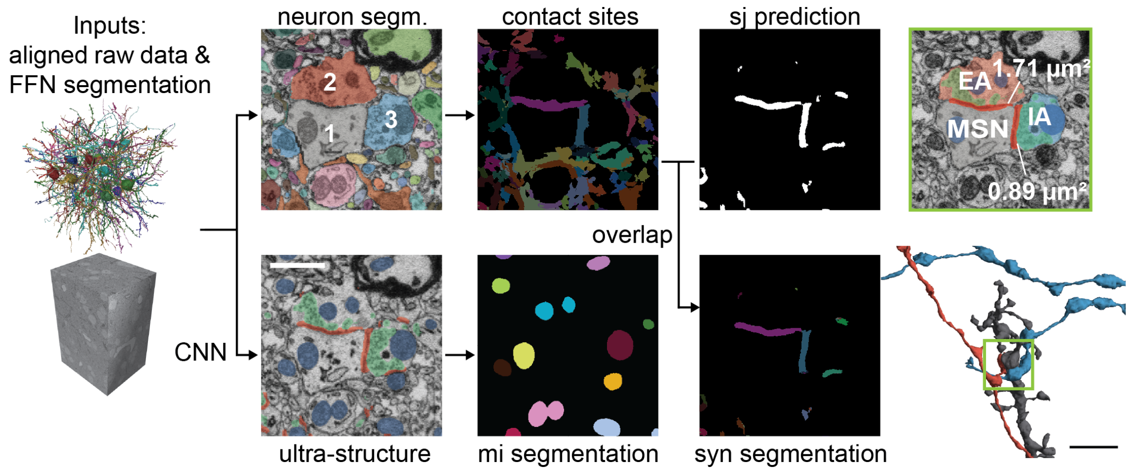


Figure 5.2: Voxel-level neuron and ultrastructure segmentation (synaptic junctions (sj) in red; mitochondria (mi) in blue; vesicle clouds (vc) in green) derived from aligned raw VEM data. While intra-cellular ultrastructure (e.g. mi segmentation) uses simple voxel-overlap, synaptic junction predictions are associated with neurons by combing the *sj* voxels with a contact site instance segmentation, which results in an instance segmentation of putative synapses (syn segmentation). The contact areas of the synapses are computed, and pre- and postsynaptic cell types assigned (1: MSN dendrite, 2: excitatory axon, EA, 3: inhibitory axon, IA). Bottom right shows a rendering of the three neurons involved in the synapse formation (green box; neuron colors correspond to the 2D overlay). Scale bars are 1 µm in the EM section and 4 µm in the neuron rendering (bottom right). Figure adapted from (Schubert et al. In review).

In contrast to Staffler et al. 2017, where features from a subvolume around the contact site were used for synapse classification, synapse reconstruction is separated in two, independent parts: firstly, the extraction of contact areas between neurons, which additionally serve for synaptic partner assignment, and secondly, the identification of synaptic junctions on voxel level (Fig. 5.2). The contact sites are combined by overlap with the prediction mask of the synaptic junctions, split by connected components into individual, putative synapse objects (Materials and Methods) which in turn receive a "synapticity" probability by a random forest classifier (10-fold cross-validation F_1-score for the synapse-true class of 0.826; precision: 0.794, recall: 0.861; *synapses-rfc*, Appendix B). Note that there is no intrinsic ordering of the neurons for the RFC partner features, because the information about pre- and postsynaptic site is not taken into account. The classifier performance was only slightly affected when duplicating the samples in the training and test splits, and swapping the partner features in the duplicates (F_1-score: 0.834; precision: 0.803, recall: 0.868).

A useful byproduct of the two-step approach are the neuron-to-neuron contacts, that would not be available if pre- and postsynaptic partners were extracted jointly (Buhmann et al. 2021; Turner et al. 2020). Ultrastructure other than synapses (mitochondria and vesicle clouds) is associated to the neuron (or neurite fragment) with which it forms the largest voxel overlap.

Instead of inferring the direction of synaptic signaling locally (Buhmann et al. 2021; Macrina et al. 2021; Shapson-Coe et al. 2021; Turner et al. 2020), the neuron skeletons were divided into subgraphs belonging to axon, dendrite, and soma based on the prediction of the morphology learning networks (Chapter 4: Learning cellular morphology). The starting point for these models are neuron representations, that include associated ultrastructure and cell meshes, as well as the cell skeleton. The compartment and cell type predictions of a neuron are then used to extend the representation of the synapses associated to it (Fig. 5.2 top right).

In order to manage the huge amount of data at a reasonable cost, huge efforts were made to ensure computational efficiency at every step of the processing. For example, instead of dense neuron processing, the employed neural morphology networks operate on sparse point clouds and all pipeline steps were implemented to be highly distributed. Processing can be roughly divided into three groups: the preparation of object property file caches (data store), contact site generation and overlay with synaptic junction predictions (synapse extraction), surface segmentation, classification and embedding of cells (morphology analysis), and the final consolidation of synapse properties with the export of a connectivity graph (synapse enrichment). A detailed overview of the intermediate steps and their dependencies is shown in Fig. C.1.

A timing of the SyConn processing on Google Cloud Platform and the *area X zebra finch, large* data set yielded overall a throughput of about 34 megavoxels per hour per CPU core and 4.4 gigavoxels per hour per GPU, which leads to an approximate cost of about \$2,000 per teravoxel of 8-bit raw VEM data at a voxel size of $10 \times 10 \times 25\,\text{nm}^3$ ($\sim$ \$800 per million cubic microns, Section 5.2).

The timed processing steps were grouped into CPU-only (data store, synapse extraction, synapse enrichment; 68.56 h processing for 1.812 teravoxels) and GPU+CPU (morphological analysis; 8.44 h point-based, 27.51 h multi-view-based processing for 1.812 teravoxels). Assuming GPU nodes only for GPU relevant processing steps, the cost per teravoxel for the different categories summed to approximately[7] \$1200 for CPU-only (1.325 hourly rate for one CPU node), \$380 for GPU+CPU (point-based; \$1200 for multi-view models; \$3.36 hourly rate for one GPU node) and \$260 for infrastructure (\$4.84 hourly rate for persistent HDD disk and \$0.348 per SSD file system server node); in total \$1840 per teravoxel (voxel size: $10 \times 10 \times 25\,\mathrm{nm}^3$).

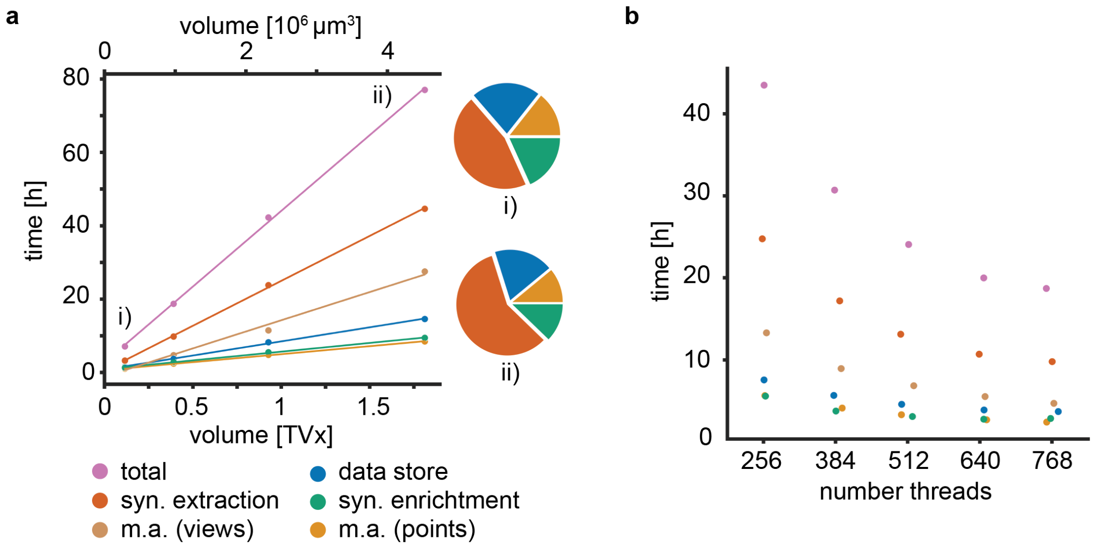


Figure 5.3: Timings of the different pipeline steps on subvolumes of *area X zebra finch, large*, grouped into synapse extraction, data store, synapse enrichment and morphology analysis (m.a.) with multi-views (views) and point clouds (points). **a** Compute time as a function of the processed volume (in teravoxels, TVx) with linear fits. Pie charts show the fraction of the different steps relative to the total time at the smallest and largest test cube (i: $0.29 \cdot 10^6 \mu\mathrm{m}^3$, syn. extraction: 0.45, data store: 0.22, syn. enrichment: 0.18, m.a. (points): 0.14; ii: $4.53 \cdot 10^6 \mu\mathrm{m}^3$, 0.58, 0.19, 0.12, 0.11). The "views" step was excluded for the "total" timings and the pie charts in i, ii. Compute resources: 24 Google Cloud computing nodes (n1-highmem-32), each with 32 virtual cores (threads), 2 Tesla P100, 208 GB memory. **b** Compute time as a function of the number of available compute nodes (8, 12, 16, 20, 24) with a constant processing volume of 0.391 teravoxels. Color code as in a. Figure adapted from (Schubert et al. In review).

The point morphology networks (Chapter 4) showed a 3.3-fold increase in speed compared to the multi-view approach (8.4 h vs. 27.5 h at 1.81 teravoxels, Fig. 5.3a), albeit with a slight performance decrease in the semantic segmentation of neuron surfaces (Section 4.3.1: Compartment prediction). Fig. 5.3a shows an increasing dominance of the synapse reconstruction (i vs. ii) and a disproportionate increase in throughput with more

[7]based on https://cloud.google.com/products/calculator

computational power (Fig. 5.3b), presumably due to the saturation of the available input/output operations of the file system.

To date, SyConn2 has been applied to three SEM data sets ranging from approximately 0.5 to 11.5 teravoxels and two species (Fig. 5.4) which is described in detail in (Kornfeld et al. 2020; Svara et al. In review).

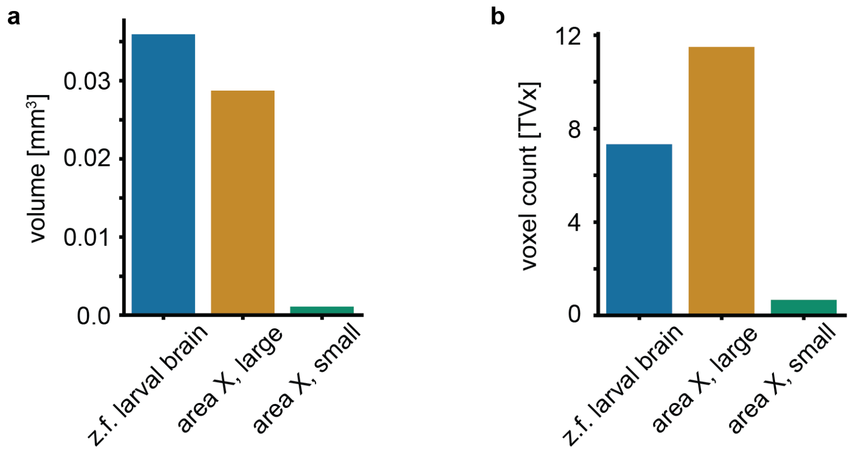


Figure 5.4: VEM data sets processed with SyConn2 (z.f.: zebra fish). **a** Volume in mm^3. **b** Volume as voxel count in teravoxels (TVx).

5.3.2 Data interface

SyConn2 was developed in Python, is open source[8] and implements an API (Application Programming Interface) that allows object properties to be accessed via key-value stores during processing and for later in-depth analyses. To ease the access of the connectome data output also for researchers that have not originally produced and analyzed the volume EM data sets, a webclient was developed based on the widely used Neuroglancer interface (Maitin-Shepard et al. 2021; Fig. 5.5). Using the SyConn client, neuroscientists can for example inspect neurons and reconstructed synapses with the associated morphology network annotations, without downloading the entire data set. This feature is intended to help a larger research community take advantage of the rapidly emerging connectomic data sets.

[8]https://github.com/StructuralNeurobiologyLab/SyConn

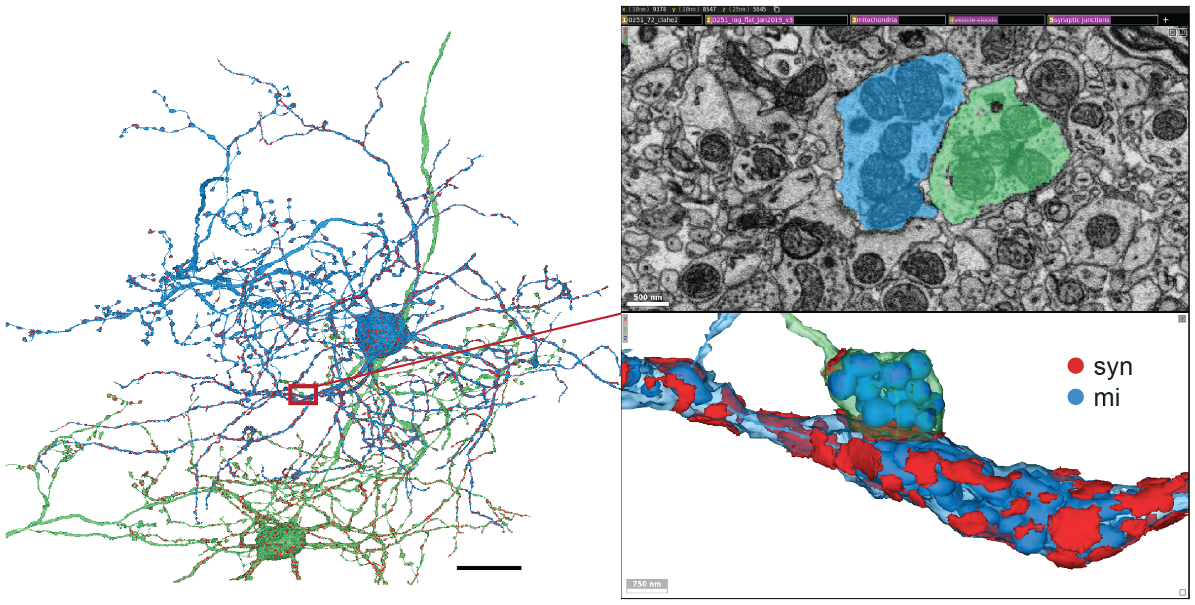


Figure 5.5: A reconstructed synapse between two pallidal-like cells (top right) in *zebra finch area X, large* visualized with the SyConn2 web interface through Neuroglancer (syn: synapse, mi: mitochondrion). Scale bar in the EM 3D rendering is 30 µm.

5.3.3 Mitochondria recruitment at synapses

In a previous analysis, the relation between the firing rates of striato-pallidal neuron classes and the mitochondrial content of different cellular compartments was analyzed (Dorkenwald, Schubert, et al. 2017). The now available, much larger pool of cell-type identified synapses allowed to test the hypothesis that larger synapses preferentially recruit mitochondria presynaptically, which could accommodate increased local energy demand (Vos et al. 2010).

To estimate the fraction of true synapses in the analysis, 52 putative synapse objects were drawn from a subset of 100 randomly selected presynaptic MSN and 38 GP cells, and annotated as true or false synapse. The annotated synapses were part of a pool of 100, which was evenly sampled from lower and upper half of the global synapse area distribution for MSN and GP respectively, e.g., 25 were presynaptic MSN and above the type-specific area median. In total, 12 GP and 13 MSN from the lower half of synapse area distributions and 13 GP and 14 MSN of the upper half were annotated. 51 synapses were found to be true, 1 upper half MSN to be incorrect.

Distances are indeed smaller between mitochondria and larger synapses, with a cell-type dependent distance distribution (Fig. 5.6a; distance to median lower half of synapses MSN: 0.833 µm, GP: 0.267 µm; median upper half MSN: 0.339 µm, GP: 0.232 µm; N synapses GP: 7,482, MSN: 59,131; p-value 0.0 for lower vs. upper half size population in both cell types using a two-sided Kolmogorov-Smirnov [9]; Materials and Methods).

In addition, pallidal neuron types (GP), which exhibit high firing rates, showed smaller, left shifted, synapse-mitochondria distance distributions compared to the sparsely firing striatal spiny neurons (MSN). Furthermore, large and small GP synapses inherited a small mitochondria-to-synapse distance, whereas mitochondria appear to be recruited selectively to large MSN synapses (Fig. 5.6a,b). This simple analysis demonstrates that queryable EM connectomic data sets with dense ultrastructural annotation provide insights that go far beyond simple connectivity analyses.

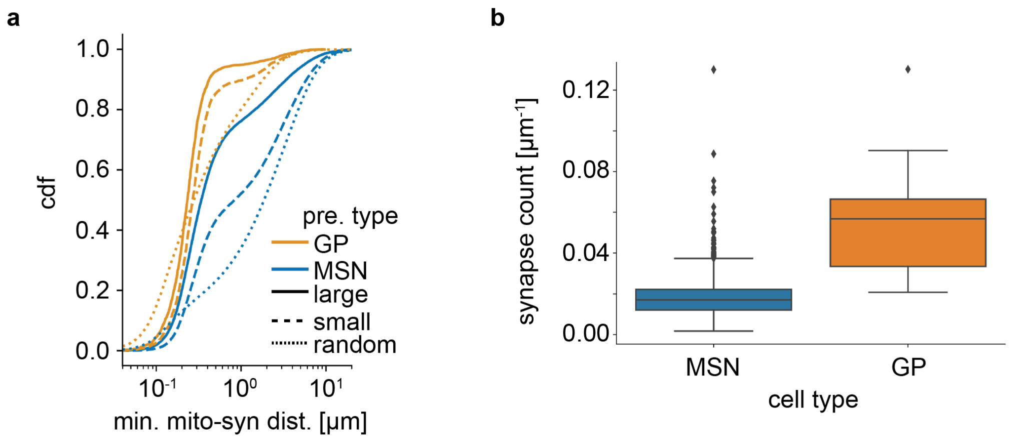


Figure 5.6: Distance analysis between synapses and mitochondria in *zebra finch area X, large*. **a** Cumulative distribution of the minimal distance between axo-dendritic synapses (and a random control, Methods) and mitochondria in GP and MSN, split into small and large synapses ($\leq$ and $>$ median of mesh area; median GP: 1.16 µm^2, MSN: 0.75 µm^2; N synapses GP: 7,482, MSN: 59,131; N random control locations for GP: 37,149, MSN: 6,128,974). The two-sided Kolmogorov-Smirnov test returned p-values of 0.0 for lower vs. upper half of the size population for GP (test statistic: 0.154) and MSN (test statistic: 0.245) and for lower vs. control for GP (test statistic: 0.195) and MSN (test statistic: 0.206). **b** Box plot (median, lower and upper quartile; whiskers, 1.5x interquartile (Q3-Q1); points, outlier) of the average synapse count per micrometer for cell types MSN (N = 6327, median: 0.017 µm^{-1}, Q1: 0.012 µm^{-1}, Q3: 0.022 µm^{-1}) and GP (N = 38, 0.057 µm^{-1}, 0.033 µm^{-1} 0.066 µm^{-1}). Two-sided Mann-Whitney U test statistic: -9.71 and p-value: $2.57 \cdot 10^{-22}$. Figure adapted from (Schubert et al. In review).

[9]`ks_2samp` method from the scipy package in "asymp" mode

5.4 Discussion

With the rapid advances in imaging pipelines, available data sets are now peaking at petabyte scales ($\sim 1\,\mathrm{mm}$; Consortium et al. 2021; Shapson-Coe et al. 2021), a trend that is likely to continue. SyConn2 contributes to the open source analysis of volume electron microscopy data, enabling community-driven development and improving reproducibility. In addition to a high degree of automation and low error rates, the cost efficiency of the analysis is a decisive factor. Excluding voxel-level segmentation maps, the cost for the extraction of a comprehensive synaptic connectivity map was estimated to be about \$800 per million cubic microns. Extrapolating this to an entire mouse brain ($5 \cdot 10^{11}\,\mu\mathrm{m}^3$, Abbott et al. 2020) yields a horrendous sum of \$400 million.

With the growing data set sizes, distributed proofreading is becoming a common tool to address remaining errors in the circuit reconstruction (C. Bishop et al. 2021; Dorkenwald et al. 2022; Zhao et al. 2018). In addition to the convenient way of visualizing detailed analysis results, the SyConn2 webclient could be extended to an interface that enables the feedback of manual corrections into the connectome database.

To lift this effort, the automatically generated cellular compartment labels could be utilized to verify neurobiological consistency in neuron representations and suggest detected inconsistencies for guided, human inspection. Such constraints are that each neuron or neurite may contain at most one soma connected component in its skeleton graph, or, if neuroanatomy permits, that dendritic and axonal neurites are only connected via a soma. These properties could be leveraged to not only identify remaining compartment prediction errors but also potential merge errors in the neuron or initial synapse segmentation. Out-of-distribution samples, such as incorrectly merged neuron fragments, might also be identifiable by exploiting the certainty of the supervised cell type classification or cluster properties in the self-supervised cell embeddings.

Chapter 6

Conclusions and Outlook

6.1 Conclusions

In the first part of this thesis, gas cluster ion beam (GCIB) milling was combined with scanning electron microscopes (SEM), resulting in a serial thick-section block-face technique for biological specimen. First imaging experiments showed the feasibility of its application with silicon wafers and a floating mask for section collection, but further advancements in section collection and during image acquisition are required for routine. The deep learning based model for autofocusing and -stigmation in SEMs showed fast convergence and reliability, making it a useful tool for the continuous and long-running image acquisitions in connectomics and beyond.

As an alternative to cell morphology analysis with 3D-EM voxel data, two types of convolutional neural networks, dubbed cellular morphology neural networks (CMNs), were adopted to exploit surface representations as a data source. The CMNs were evaluated on the tasks of cell type classification and surface segmentation of neuronal compartments and achieved high performance scores. In addition, a self-supervised training paradigm was applied to extract morphological features for neuron clustering that showed qualitative agreement with the supervised cell type classifier.

The developed synaptic connectivity framework SyConnn2 fundamentally upgraded the previous version (Dorkenwald et al. 2017). It, within the limits of ground truth generation, automates the steps of synaptic connectivity inference, cell type and compartment classification and was applied to EM data sets with up to 12 teravoxels. The framework thus substantially improves the degree of automation in synaptic circuit analysis and allows researchers to inspect and query the extracted connectivity results.

6.2 Outlook

The high removal rate of GCIB milling enables future combination with a high-throughput multi-beam SEM, but reliable section collection remains challenging, albeit critical for the use of GCIB-SEM in volume electron microscopy (VEM). Whether the proposed collection procedure with floating masks can be improved to achieve the reliability required for volume electron microscopy acquisitions remains to be determined.

The DeepFocus approach will be a useful tool for upcoming acquisitions with GCIB-SEM, which require frequent refocusing. Also, the adoption to other microscope types or the control of more machine parameters seem plausible.

Surface meshes offer a lightweight but expressive neuron representation, which can be exploited by deep learning models, such as the here presented cellular morphology networks. Given the data sparsity and ease of transferability between data sets with inputs not directly relying on sometimes varying image statistics, mesh-based approaches seem worthwhile for VEM. It remains to be tested if CMNs profit from other types of neural network architectures. MeshCNN (Hanocka et al. 2019), for example, takes into account the edges of the mesh representations in addition to point locations.

With automated circuit reconstruction, the time required to analyze EM data sets is substantially reduced. As a result, connectomic studies targeting brain region development become feasible.

Appendix A

Biological samples

The following samples and data were used for development, testing and analysis:

- *zebra finch area X, small* Part of the zebra finch nucleus area X provided by J. Kornfeld. Data set size: 10,664 x 10,914 × 5701 8 bit voxels of size 9 x 9 x 20 nm. Neuron segmentation, skeletons and meshes were provided by Google Research.
- *zebra finch area X, large* Part of the zebra finch nucleus area X provided by J. Kornfeld. Data set size: 27,119 x 27,350 × 15,494 8 bit voxels of size 10 x 10 x 25 nm. Neuron, ultrastructure and synapse type segmentation was provided by Google Research.

Appendix B

Ground truth data

If not otherwise stated, data sets were generated in *zebra finch area X, large*.

- *semseg-fine-train* Five node-wise annotated cell reconstructions (*zebra finch area X, small*). Labels used: Dendritic shaft, spine neck, spine head, other (axon/soma); Number of annotated nodes: 21,081 (dendritic shaft), 3223 (neck), 5601 (head), 3245 (other). The multi-view data was split into training (24,248 views) and validation (6062 views); views of all reconstructions were shuffled prior to the split. Note that here the rendering location sampling was performed using the center vertex within 2 µm voxels. Multi-views contained $N = 5$ projections. Label boundaries were smoothed by assigning each vertex the majority label of 40 vertices, that were found by BFS on a vertex graph. All cell mesh vertices were used as graph nodes and edges were added between vertices that were within a distance of 120 nm.

- *semseg-fine-test-vertices*: One dendritic arbor (*zebra finch area X, small*). Number of annotated nodes: 75 (head), 122 (neck), 770 (shaft).

- *dendritic-synapses*: Four neurons (*zebra finch area X, small*) with synapses manually annotated into spine head or dendritic shaft (shaft: 94, head: 88).

- *semseg-coarse-train*: 45 node-wise annotated neurons (*zebra finch area X, small*). Number of annotated nodes: 94,984 (axon), 112,103 (dendrite), 102,324 (soma), 19,639 (bouton *en passant*) and 5745 (terminal bouton). Views of all reconstructions were shuffled prior to splitting into train (90%) and valid (10%) data set. Multi-views contained $N = 4$ projections.

- *semseg-coarse-test*: 6 node-wise annotate neurons (*zebra finch area X, small*). Number of annotated nodes: 27,405 (dendrite), axon (3812), soma (11,551), bouton *en-passant* (3093), terminal bouton (1210).

- *semseg-large-train*: 29 sparsely, node-wise annotated neurons (6 DA, 6 HVC, 2 LMAN, 1 FS, 2 GPe, 2 GPi, 2 LTS, 2 MSN, 2 NGF, 2 STN and 2 TAN) with the following vertex and node support: dendrite (2848956, 41008), axon (2908024, 92318), soma (113642, 83), bouton *en-passant* (1423132, 23381), terminal bouton (218560, 3063), neck (128923, 6518), head (176322, 5506).

- *semseg-large-test*: 13 sparsely, node-wise annotated neurons (2 DA, 2 HVC and 1 of each of the other 9 cell types, see below) with the following vertex and node support: dendrite (1297702, 21037), axon (1720652, 60872), soma (766222, 3647), bouton *en-passant* (1069485, 17401), terminal bouton (137827, 2068), neck (70612, 4264), head (77328, 2667).

- *celltypes-large*: 253 neuron reconstructions labeled as one of 11 classes. Class support: STN (35), DA (19), MSN (32), LMAN (31), HVC (33), TAN (12), GPe (14), GPi (17), FS (27), LTS (10), NGF (23).

- *synapses-rfc*: 300 neuron-neuron contact locations, manually annotated into synaptic (156) and non-synaptic (144).

- *myelin-gt*: Three manually proofread cubes from *zebra finch area X, small*, with a total of approx. 100 megapixels (voxel size [µm]: 36, 36, 80). Original prediction was based on the output of the inner myelin model presented in (Dorkenwald et al. 2017), i.e. foreground pixels flag the entire volume of a myelinated axon. Class support: background (98%), foreground (2%).

- *synapse-type-gt*: Based on from *zebra finch area X, small*, four manually annotated cubes and 900 small cubes that were auto-generated from synapse locations between cell types with known synapse type (background: 95.8%, asymmetric: 1.5%, symmetric: 2.7%). For training patch creation, cubes were drawn based on their total volume fraction.

Appendix C

SyConn2 flowchart

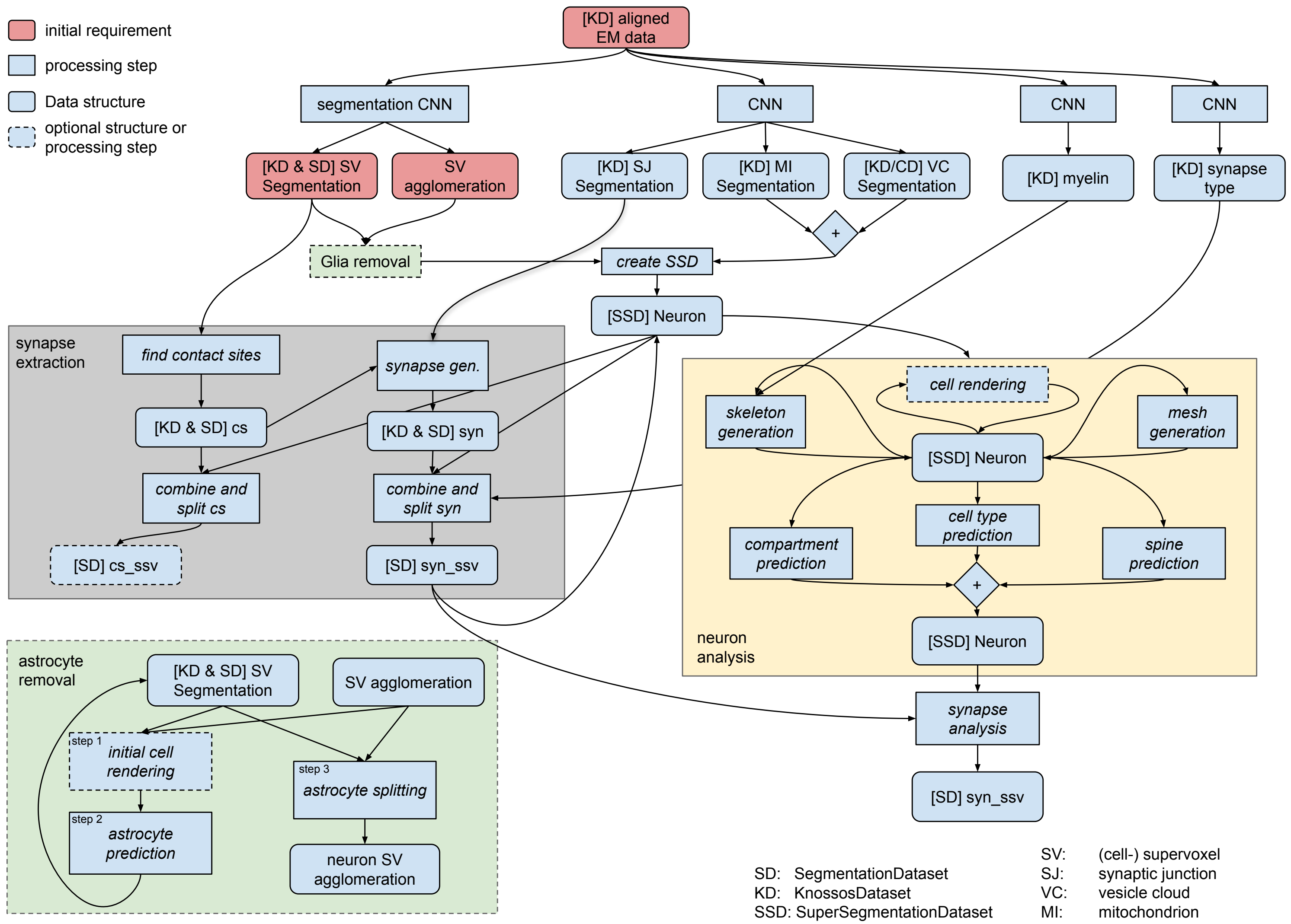


Figure C.1: Flowchart of the SyConn2 architecture.

Appendix D

List of Abbreviations

AF	auto-focus
BFS	breadth-first search
CMN	cellular morphology neural network
CNN	convolutional neural network
CPU	central processing unit
DA	putative dopaminergic axon
EM	electron microscopy
FCN	fully convolutional network
FFN	flood-filling neural network
FIB	focused ion beam
FOV	field of view
FS	putative fast-spiking neuron
GB	gigabyte
GCIB	gas cluster ion beam
GPe	putative pallidal-like neuron of direct pathway
GPi	putative pallidal-like neuron of indirect pathway
GPU	graphics processing unit
GUI	graphical user interface

HPC high-performance computing

HVC putative HVC-projection axon

k-NN k-nearest neighbors

LMAN putative LMAN-projection axon

LTS putative low-threshold-spiking neuron

MAE mean absolute error

MLP multilayer perceptron

MSEM multi-beam scanning electron microscope

MSN putative medium spiny neuron

NGF putative neuro glia form

OLS ordinary least squares

PCA principal component analysis

RAM random-access memory

RF random forest

ROI region of interest

s.d. standard deviation

SBEM serial block-face scanning electron microscopy

SEM scanning electron microscope

SNR signal-to-noise ratio

STN putative subthalamic-nucleus-like neuron

TAN putative tonically active neuron

TEM transmission electron microscope

VEM volume electron microscopy

Bibliography

Abbott, L. F., Bock, D. D., Callaway, E. M., Denk, W., ... Van Essen, D. C. (2020). The mind of a mouse. *Cell*, *182*, 1372–1376.

Bartol, J., Thomas M, Bromer, C., Kinney, J., Chirillo, M. A., ... Sejnowski, T. J. (2015). Nanoconnectomic upper bound on the variability of synaptic plasticity. *eLife*, *4*, e10778.

Batten, C. F. (2000). *Autofocusing and astigmatism correction in the scanning electron microscope* (Master's thesis). University of Cambridge.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Commun. ACM*, *18*, 509–517.

Berman, M., Triki, A. R., & Blaschko, M. B. (2018). The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 4413–4421.

Binding, J., Mikula, S., & Denk, W. (2013). Low-dosage Maximum-A-Posteriori focusing and stigmation. *Microsc. Microanal.*, *19*, 38–55.

Bishop, C., Matelsky, J., Wilt, M., Downs, J., ... Gray-Roncal, W. (2021). Confirms: A toolkit for scalable, black box connectome assessment and investigation. *2021 43rd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2444–2450.

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Springer-Verlag.

Boulch, A. (2020). Convpoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, *88*, 24–34.

Boulch, A., Guerry, J., Le Saux, B., & Audebert, N. (2018). SnapNet: 3D point cloud semantic labeling with 2d deep segmentation networks. *Computers & Graphics*, *71*, 189–198.

Boulch, A., Puy, G., & Marlet, R. (2020). Fkaconv: Feature-kernel alignment for point cloud convolution. *Proceedings of the Asian Conference on Computer Vision*, *12622*, 381–399.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32.

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

Budzillo, A., Duffy, A., Miller, K. E., Fairhall, A. L., & Perkel, D. J. (2017). Dopaminergic modulation of basal ganglia output through coupled excitation–inhibition. *Proceedings of the National Academy of Sciences*, *114*, 5713–5718.

Buhmann, J., Sheridan, A., Malin-Mayor, C., Schlegel, P., … Lee, W.-C. A., et al. (2021). Automatic detection of synaptic partners in a whole-brain drosophila electron microscopy data set. *Nature Methods*, *18*, 771–774.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., … Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information*, *11*, 125.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *Proceedings of the 37th International Conference on Machine Learning*, *119*, 1597–1607.

Ciresan, D. C., Giusti, A., Gambardella, L. M., & Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in Neural Information Processing Systems*, *25*, 2852–2860.

Consortium, M., Bae, J. A., Baptiste, M., Bodor, A. L., … Yu, S.-c. (2021). Functional connectomics spanning multiple areas of mouse visual cortex. *bioRxiv*, 2021.07.28.454025.

Costa, M., Manton, J. D., Ostrovsky, A. D., Prohaska, S., & Jefferis, G. S. (2016). Nblast: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases. *Neuron*, *91*, 293–311.

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning.* Cambridge University Press.

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

Dorkenwald, S., McKellar, C. E., Macrina, T., Kemnitz, N., … Seung, H. S. (2022). Flywire: Online community for whole-brain connectomics. *Nature Methods*, *19*, 119–128.

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

Dorkenwald, S., Turner, N. L., Macrina, T., Lee, K., … Seung, H. S. (2021). Binary and analog variation of synapses between cortical pyramidal neurons. *bioRxiv*, 2019.12.29.890319.

Eberle, A. L., Mikula, S., Schalek, R., Lichtman, J., … Zeidler, D. (2015). High-resolution, high-throughput imaging with a multibeam scanning electron microscope. *J. Microsc.*, *259*, 114–120.

Egerton, R. F. (2005). *Physical principles of electron microscopy* (Vol. 56). Springer.

Egerton, R. F., Li, P., & Malac, M. (2004). Radiation damage in the TEM and SEM [International Wuhan Symposium on Advanced Electron Microscopy]. *Micron*, *35*, 399–409.

Erasmus, S., & Smith, K. (1982). An automatic focusing and astigmatism correction system for the SEM and CTEM. *Journal of Microscopy*, *127*, 185–199.

Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, *57*, 238–247.

Fukushima, K., & Miyake, S. (1982). Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, *15*, 455–469.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, *9*, 249–256.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323.

Goldstein, J. I., Newbury, D. E., Joy, D. C., Lyman, C. E., ... Michael, J. R. (2003). *Scanning electron microscopy and x-ray microanalysis*. Springer.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press.

Gour, A., Boergens, K. M., Heike, N., Hua, Y., ... Helmstaedter, M. (2021). Postnatal connectomic development of inhibition in mouse barrel cortex. *Science*, *371*, eabb4534.

Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, *37*, 362–386.

Haberl, M. G., Churas, C., Tindall, L., Boassa, D., ... Peltier, S. T., et al. (2018). Cdeep3m—plug-and-play cloud-based deep learning for image segmentation. *Nature Methods*, *15*, 677–680.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., ... Cohen-Or, D. (2019). MeshCNN: A network with an edge. *ACM Trans. Graph.*, *38*, 90:1–90:12.

Hayworth, K. J., Kasthuri, N., Schalek, R., & Lichtman, J. W. (2006). Automating the collection of ultrathin serial sections for large volume TEM reconstructions. *Microscopy and Microanalysis*, *12*, 86–87.

Hayworth, K. J., Peale, D., Januszewski, M., Knott, G. W., ... Hess, H. F. (2020). Gas cluster ion beam SEM for imaging of large tissue samples with 10 nm isotropic resolution. *Nature Methods*, *17*, 68–71.

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

Heinrich, L., Bennett, D., Ackerman, D., Park, W., ... Xu, C. S., et al. (2021). Whole-cell organelle segmentation in volume electron microscopy. *Nature*, *599*, 141–146.

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

Helmstaedter, M., Briggman, K. L., & Denk, W. (2011). High-accuracy neurite reconstruction for high-throughput neuroanatomy. *Nat. Neurosci.*, *14*, 1081–1088.

Heymann, J. A., Hayles, M., Gestmann, I., Giannuzzi, L. A., ... Subramaniam, S. (2006). Site-specific 3D imaging of cells and tissues with a dual beam microscope. *Journal of Structural Biology*, *155*, 63–73.

Holler, S., Köstinger, G., Martin, K. A., Schuhknecht, G. F., & Stratford, K. J. (2021). Structure and function of a neocortical synapse. *Nature*, *591*, 111–116.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*, 359–366.

Horstmann, H., Körber, C., Sätzler, K., Aydin, D., & Kuner, T. (2012). Serial section scanning electron microscopy (s3em) on silicon wafers for ultra-structural volume imaging of cells and tissues. *PLOS ONE*, *7*, 1–8.

Hu, Q., Yang, B., Xie, L., Rosa, S., … Markham, A. (2020). RandLA-Net: Efficient semantic segmentation of large-scale point clouds. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 11105–11114.

Hua, Y., Laserstein, P., & Helmstaedter, M. (2015). Large-volume en-bloc staining for electron microscopy-based connectomics. *Nature Communications*, *6*, 1–7.

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, *18*, 6869–6898.

Hubel, D. H., & Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*, 106.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, *37*, 448–456.

Jain, V., Murray, J. F., Roth, F., Turaga, S. C., … Seung, H. S. (2007). Supervised learning of image restoration with convolutional networks. *International Conference on Computer Vision (ICCV)*, 1–8.

Januszewski, M., Kornfeld, J., Li, P. H., Pope, A., … Jain, V. (2018). High-precision automated reconstruction of neurons with flood-filling networks. *Nature Methods*, *15*, 605–610.

Johnson, E. C., Wilt, M., Rodriguez, L. M., Norman-Tenazas, R., … Downs, J., et al. (2020). Toward a scalable framework for reproducible processing of volumetric, nanoscale neuroimaging datasets. *GigaScience*, *9*, giaa147.

Kanaya, K., & Okayama, S. (1972). Penetration and energy-loss theory of electrons in solid targets. *Journal of Physics D: Applied Physics*, *5*, 43–58.

Kar, M. K., Nath, M. K., & Neog, D. R. (2021). A review on progress in semantic image segmentation and its application to medical images. *SN Computer Science*, *2*, 1–30.

Kasthuri, N., Hayworth, K. J., Berger, D. R., Schalek, R. L., … Lichtman, J. W. (2015). Saturated reconstruction of a volume of neocortex. *Cell*, *162*, 648–661.

Kim, H., Oh, M., Lee, H., Jang, J., … Lee, J. (2019). Deep-learning based autofocus score prediction of scanning electron microscope. *Microscopy and Microanalysis*, *25*, 182–183.

Kim, J. S., Greene, M. J., Zlateski, A., Lee, K., … EyeWirers. (2014). Space-time wiring specificity supports direction selectivity in the retina. *Nature*, *509*, 331–336.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations (ICLR)*.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Klimesch, J. (2020). *Analysis of neuronal morphology using semantic segmentation of point clouds* (BA thesis). Technische Universität München.

Knoll, M., & Ruska, E. (1932). Das elektronenmikroskop. *Zeitschrift für Physik*, *78*, 318–339.

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

Kornfeld, J. (2018). *Connectomic analyses in the zebra finch brain* (Doctoral dissertation). Ruperto-Carola University of Heidelberg.

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

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

Kornfeld, J., Januszewski, M., Schubert, P. J., Jain, V., ... Fee, M. (2020). An anatomical substrate of credit assignment in reinforcement learning. *bioRxiv*, 2020.02.18.954354.

Köthe, U. (2000). *Generische programmierung für die bildverarbeitung* (Doctoral dissertation). University of Hamburg.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1106–1114.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., ... Jackel, L. D. (1989). Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, *2*, 396–404.

Lee, K., Lu, R., Luther, K., & Seung, H. S. (2021). Learning and segmenting dense voxel embeddings for 3D neuron reconstruction. *IEEE Transactions on Medical Imaging*, *40*, 3801–3811.

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

Lee, W., Nam, H. S., Kim, Y. G., Kim, Y. J., ... Yoo, H. (2021). Robust autofocusing for scanning electron microscopy based on a dual deep learning network. *Scientific Reports*, *11*, 1–12.

Leighton, S. B. (1981). SEM images of block faces, cut by a miniature microtome within the SEM-a technical note. *Scanning electron microscopy*, 73–76.

Li, C., Moatti, A., Zhang, X., Ghashghaei, H. T., & Greenabum, A. (2021). Deep learning-based autofocus method enhances image quality in light-sheet fluorescence microscopy. *Biomedical optics express*, *12*, 5214–5226.

Li, H., Januszewski, M., Jain, V., & Li, P. H. (2020). Neuronal subcompartment classification and merge error correction. *Medical Image Computing and Computer Assisted Intervention*, *12265*, 88–98.

Li, Y., Bu, R., Sun, M., Wu, W., ... Chen, B. (2018). PointCNN: Convolution on X-transformed points. *Advances in Neural Information Processing Systems*, *31*, 828–838.

Liu, X., Deng, Z., & Yang, Y. (2019). Recent progress in semantic image segmentation. *Artificial Intelligence Review*, *52*, 1089–1106.

Lorensen, W. E., & Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Comput. Graph.*, *21*, 163–169.

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *7th International Conference on Learning Representations (ICLR)*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Macrina, T., Lee, K., Lu, R., Turner, N. L., . . . Seung, H. S. (2021). Petascale neural circuit reconstruction: Automated methods. *bioRxiv*, 2021.08.04.455162.

Maitin-Shepard, J., Baden, A., Silversmith, W., Perlman, E., . . . Li, P. H. (2021). *Neuroglancer*. Zenodo. https://doi.org/10.5281/zenodo.5573293

Mann, H. B., & Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, *18*, 50–60.

McInnes, L., Healy, J., & Melville, J. (2020). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv:1802.03426*.

Moe, S., Rustad, A. M., & Hanssen, K. G. (2018). Machine learning in control systems: An overview of the state of the art. *International Conference on Artificial Intelligence (AI)*, *11311*, 250–265.

Motta, A., Berning, M., Boergens, K. M., Staffler, B., . . . Helmstaedter, M. (2019). Dense connectomic reconstruction in layer 4 of the somatosensory cortex. *Science*.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*, 308–313.

Odena, A., Dumoulin, V., & Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*, *1*, e3.

Otter, D. W., Medina, J. R., & Kalita, J. K. (2020). A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, *32*, 604–624.

Paszke, A., Gross, S., Massa, F., Lerer, A., . . . Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*, 8024–8035.

Paxman, R. G., Schulz, T. J., & Fienup, J. R. (1992). Joint estimation of object and aberrations by using phase diversity. *JOSA A*, *9*, 1072–1085.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). PointNet: Deep learning on point sets for 3D classification and segmentation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 77–85.

Qi, C. R., Su, H., Nießner, M., Dai, A., . . . Guibas, L. J. (2016). Volumetric and multi-view CNNs for object classification on 3D data. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 5648–5656.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, *30*, 5099–5108.

Ramachandran, P., Zoph, B., & Le, Q. V. (2018). Searching for activation functions. *6th International Conference on Learning Representations (ICLR).*

Ramón y Cajal, S. (1888). Estructura de los centros nerviosos de las aves. *Rev. Trim. Histol. Norm. Pat*, *1*, 1–10.

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of adam and beyond. *6th International Conference on Learning Representations (ICLR).*

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention*, *9351*, 234–241.

Rudnaya, M. (2011). *Automated focusing and astigmatism correction in electron microscopy* (Doctoral dissertation). Technische Universiteit Eindhoven.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Sage, D., & Unser, M. (2003). Teaching image-processing programming in Java. *IEEE Signal Processing Magazine*, *20*, 43–52.

Sato, M., Bitter, I., Bender, M. A., Kaufman, A. E., & Nakajima, M. (2000). TEASAR: tree-structure extraction algorithm for accurate and robust skeletons. *8th Pacific Conference on Computer Graphics and Applications, PG 2000, Hong Kong, October 3-5, 2000*, 281.

Schalek, R., Kasthuri, N., Hayworth, K., Berger, D., . . . Lichtman, J. (2011). Development of high-throughput, high-resolution 3D reconstruction of large-volume biological tissue using automated tape collection ultramicrotomy and scanning electron microscopy. *Microscopy and Microanalysis*, *17*, 966–967.

Scheffer, L. K., Xu, C. S., Januszewski, M., Lu, Z., . . . Plaza, S. M. (2020). A connectome and analysis of the adult central brain. *Elife*, *9*, e57443.

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., . . . Schmid, B., et al. (2012). Fiji: An open-source platform for biological-image analysis. *Nature Methods*, *9*, 676–682.

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

Schneider-Mizell, C. M., Gerhard, S., Longair, M., Kazimiers, T., . . . Cardona, A. (2016). Quantitative neuroanatomy for connectomics in *Drosophila*. *eLife*, *5*, e12059.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 815–823.

Schubert, P. J. (2017). *Cellular morphology learning neural networks* (Master's thesis). Ruperto-Carola University of Heidelberg.

Schubert, P. J., Dorkenwald, S., Januszewski, M., Klimesch, J., . . . Kornfeld, J. (In review). *SyConn2: Dense synaptic connectivity inference for volume EM.*

Schubert, P. J., Dorkenwald, S., Januszewski, M., Jain, V., & Kornfeld, J. (2019). Learning cellular morphology with neural networks. *Nature Communications*, *10*, 1–12.

Schubert, P. J., & Kornfeld, J. (2021, December 2). *Method for automatic focusing and astigmatism correction for an electron microscope* (European pat.) [prending].

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. *Proceedings of the 9th Python in Science Conference*, *57*, 61.

Shapson-Coe, A., Januszewski, M., Berger, D. R., Pope, A., … Lichtman, J. W. (2021). A connectomic study of a petascale fragment of human cerebral cortex. *bioRxiv*, 2021.05.29.446289.

Shelhamer, E., Long, J., & Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, *39*, 640–651.

Sheridan, A., Nguyen, T., Deb, D., Lee, W.-C. A., … Funke, J. (2021). Local shape descriptors for neuron segmentation. *bioRxiv*, 2021.01.18.427039.

Silversmith, W., Bae, J. A., Li, P. H., & Wilson, A. (2021). *Kimimaro: Skeletonize densely labeled 3D image segmentations.* https://doi.org/10.5281/zenodo.5539913

Simard, P., Steinkraus, D., & Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, 958–963.

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations (ICLR).*

Sofroniew, N., Lambert, T., Evans, K., Nunez-Iglesias, J., … Har-Gil, H. (2021). *Napari.* Zenodo. https://doi.org/10.5281/zenodo.3555620

Sommer, C., Straehle, C., Köthe, U., & Hamprecht, F. A. (2011). Ilastik: Interactive learning and segmentation toolkit. *IEEE International Symposium on Biomedical Imaging (ISBI)*, 230–233.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & 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., … Helmstaedter, M. (2017). SynEM, automated synapse detection for connectomics. *eLife*, *6*, e26414.

Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. G. (2015). Multi-view convolutional neural networks for 3D shape recognition. *International Conference on Computer Vision (ICCV)*, 945–953.

Sudre, C. H., Li, W., Vercauteren, T., Ourselin, S., & Cardoso, M. J. (2017). Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (DLMIA)*, *10553*, 240–248.

Svara, F., Förster, D., Kubo, F., Januszewski, M., … Baier, H. (In review). Automated synapse-level reconstruction of neural circuits in the larval zebrafish brain.

Templier, T. (2019). MagC, magnetic collection of ultrathin sections for volumetric correlative light and electron microscopy. *eLife*, *8*, e45696.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., . . . Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. *International Conference on Computer Vision (ICCV)*, 6410–6419.

Titze, B., Genoud, C., & Friedrich, R. W. (2018). SBEMimage: Versatile acquisition control software for serial Block-Face electron microscopy. *Front. Neural Circuits*, *12*, 54.

Turaga, S. C., Murray, J. F., Jain, V., Roth, F., . . . Seung, H. S. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Comput.*, *22*, 511–538.

Turner, N. L., Lee, K., Lu, R., Wu, J., . . . Seung, H. S. (2020). Synaptic partner assignment using attentional voxel association networks. *International Symposium on Biomedical Imaging (ISBI)*, 1–5.

Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., . . . Yu, T. (2014). Scikit-image: Image processing in python. *PeerJ*, *2*, e453.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., . . . SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272.

von Ardenne, M. (1938). Das Elektronen-Rastermikroskop. *Zeitschrift für Physik*, *109*, 553–572.

Vos, M., Lauwers, E., & Verstreken, P. (2010). Synaptic mitochondria in synaptic transmission and organization of vesicle pools in health and disease. *Frontiers in synaptic neuroscience*, *2*, 139.

Wanner, A. A., Genoud, C., & Friedrich, R. W. (2016). 3-dimensional electron microscopic imaging of the zebrafish olfactory bulb and dense reconstruction of neurons. *Scientific data*, *3*, 1–15.

White, J. G., Southgate, E., Thomson, J. N., & Brenner, S. (1986). The structure of the nervous system of the nematode caenorhabditis elegans. *Philosophical Transactions of the Royal Society of London*, *314*, 1–340.

Witvliet, D., Mulcahy, B., Mitchell, J. K., Meirovitch, Y., . . . Holmyard, D., et al. (2021). Connectomes across development reveal principles of brain maturation. *Nature*, *596*, 257–261.

Wu, W., Qi, Z., & Li, F. (2019). PointConv: Deep convolutional networks on 3D point clouds. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 9621–9630.

Wu, Y., & He, K. (2018). Group normalization. *Proceedings of the European conference on computer vision (ECCV)*, *11217*, 3–19.

Wu, Z., Song, S., Khosla, A., Yu, F., . . . Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 1912–1920.

Xiang, T., Zhang, C., Song, Y., Yu, J., & Cai, W. (2021). Walk in the cloud: Learning curves for point clouds shape analysis. *International Conference on Computer Vision (ICCV)*, 915–924.

Xu, C. S., Hayworth, K. J., Lu, Z., Grob, P., . . . Hess, H. F. (2017). Enhanced FIB-SEM systems for large-volume 3D imaging. *eLife*, *6*, e25916.

Yang, H. J., Oh, M., Jang, J., Lyu, H., & Lee, J. (2020). Robust deep-learning based autofocus score prediction for scanning electron microscope. *Microscopy and Microanalysis*, *26*, 702–705.

Yang, S. J., Berndl, M., Ando, D. M., Barch, M., ... Rueden, C. T., et al. (2018). Assessing microscope image focus quality with deep learning. *BMC bioinformatics*, *19*, 1–9.

Yin, W., Brittain, D., Borseth, J., Scott, M. E., ... da Costa, N. M. (2020). A petascale automated imaging pipeline for mapping neuronal circuits with high-throughput transmission electron microscopy. *Nature Communications*, *11*, 1–12.

Yoo, A. B., Jette, M. A., & Grondona, M. (2003). SLURM: simple linux utility for resource management. *Job Scheduling Strategies for Parallel Processing (JSSPP)*, *2862*, 44–60.

Zeng, A., Song, S., Lee, J., Rodriguez, A., & Funkhouser, T. (2020). Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, *36*, 1307–1319.

Zhao, T., Olbris, D. J., Yu, Y., & Plaza, S. M. (2018). Neutu: Software for collaborative, large-scale, segmentation-based connectome reconstruction. *Frontiers in Neural Circuits*, *12*, 101.

Zheng, Z., Lauritzen, J. S., Perlman, E., Robinson, C. G., ... Sharifi, N., et al. (2018). A complete electron microscopy volume of the brain of adult drosophila melanogaster. *Cell*, *174*, 730–743.

Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.

Acknowledgements

First of all, I want to thank Winfried Denk for his supervision and support throughout the years, without him this work would not have been possible. I greatly thank my co-advisor Jörgen Kornfeld for many, very productive discussions and intensive feedback. The past years have been an unforgettable and adventurous time in many positive ways.

I want to thank the entire team of the Denk department and Kornfeld laboratory for a great working environment, in particular Alexandra Rother, Maria Kormacheva and Rangoli Saxena and Jonas Hemesath for helpful discussions and their expertise.

I would like to thank my students Jonathan Klimesch (Bachelor's Thesis), Andrei Mancu (Bachelor's Thesis), Christina Schick (Bachelor's Thesis) and Hashir Ahmad (Master's theis) for working with me on their thesis and many inspiring discussions.

I want to thank the student assistants Christina Schick, Anna Umurzakova, David Nagel and Julian Hendricks for annotating neurons and EM images, and Martin Drawitsch for maintaining and continuously improving elektronn3 neural network library.

Furthermore, I thank Anton Grüner (Zeiss) for his technical support and advice on the electron microscopes.

I thank Alexandra Rother for the staining and embedding of tissue samples.

I would like to thank Gavyn Trowbridge (Ion Optika) and Maria Kormacheva for assistance with the ion gun.

I thank Jörgen Kornfeld and Fabian Svara for realizing KNOSSOS and the whole K-team for support and new features, in particular Norbert Pfeiler and My-Tien Nguyen.

Additionally, I am very grateful for the funding from the Max Planck Society.

I would like to thank Viren Jain (Google Research) for providing Google Cloud resources.

I also want to thank Sven Dorkenwald, Jennifer Krebs and Franz Rieger for comments on the manuscript.

I would like to thank Lorenz Hüdepohl and Christian Guggenberger from the MPCDF for compute cluster support, and Jürgen Tritthardt and Stefan Apel for their outstanding support with the electron microscope setups and help with designing and creating many parts together with the workshop.

Ich danke meiner gesamten Familie, sowie der Familie Reich für beständigen, reichhaltigen Zuspruch.

Finally, I want to thank Charlotte for her unwavering support, energy and love over all these years, and the years to come.