DISSERTATION
submitted
to the
Combined Faculty for the Natural
Sciences and Mathematics
of
Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Put forward by
MSc Philipp Hanslovsky
Born in Ludwigsburg, Germany
Oral examination:

Isotropic Reconstruction of Neural Morphology from Large Non-Isotropic 3D Electron Microscopy

Philipp Hanslovsky

Advisor: Prof. Dr. Fred A. Hamprecht

ABSTRACT

Neuroscientists are increasingly convinced that it is necessary to reconstruct the precise wiring and synaptic connectivity of biological nervous systems to eventually decipher their function. The urge to reconstruct ever larger and more complete synaptic wiring diagrams of animal brains has created an entire new subfield of neuroscience: *Connectomics*. The reconstruction of connectomes is difficult because neurons are both large and small. They project across distances of many millimeters but each individual neurite can be as thin as a few tens of nanomaters. In order to reconstruct all neurites in densely packed neural tissues, it is necessary to image this tissue at nanometer resolution which, today, is only possible with 3D electron microscopy (3D-EM).

Over the last decade, 3D-EM has become significantly more reliable than ever before. Today, it is possible to routinely image volumes of up to a cubic millimeter, covering the entire brain of small model organisms such as that of the fruit fly *Drosophila melanogaster*. These volumes contain tens or hundreds of tera-voxels and cannot be analyzed manually. Efficient computational methods and tools are needed for all stages of connectome reconstruction: (1) assembling distortion and artifact free volumes from serial section EM, (2) precise automatic reconstruction of neurons and synapses, and (3) efficient and user-friendly solutions for visualization and interactive proofreading. In this dissertation, I present new computational methods and tools that I developed to address previously unsolved problems covering all of the above mentioned aspects of EM connectomics.

In chapter 2, I present a new method to correct for planar and non-planar axial distortion and to sort unordered section series. This method was instrumental for the first ever acquisition of a complete brain of an adult *Drosophila melanogaster* imaged with 3D-EM.

Machine learning, in particular deep learning, and the availability of public training and test data has had tremendous impact on the automatic reconstruction of neurons and synapses from 3D-EM. In chapter 3, I present a novel artificial neural network architecture that predicts neuron boundaries at quasi-isotropic resolution from non-isotropic 3D-EM. The goal is to create a high-quality oversegmentation with large three-dimensional fragments for faster manual proofreading.

In chapter 4, I present software libraries and tools that I developed to support the processing, visualization, and analysis of large 3D-EM data and connectome reconstructions. Using this software, we generated the largest currently existing training and test data for connectome reconstruction from non-isotropic 3D-EM. I will particularly emphasize my flexible interactive proof-reading tool Paintera that I built on top of the libraries and tools that I have developed over the last four years.

Neurowissenschaftler sind zunehmend davon überzeugt, dass es notwendig ist, die synaptische Verbindung und den „Schaltplan“ biologischer Nervensysteme präzise zu rekonstruieren, um deren Funktion zu entschlüsseln. Die Dringlichkeit, immer größere und vollständigere synaptische Schaltpläne zu rekonstruieren, hat eine gänzlich neue Disziplin der Neurowissenschaften begründet: Konnektomiks. Die Rekonstruktion von Konnektomen ist deshalb schwer, weil Nervenzellen zugleich sehr groß und sehr klein sind. Eine einzelne Zelle kann sich über mehrere Millimeter erstrecken, während einzelne Neuriten dünner als hundert Nanometer sind. Um alle Neuriten in dichtgepacktem neuronalen Gewebe vollständig zu rekonstruieren, ist es notwendig, das Gewebe mit einer Auflösung von wenigen Nanometern pro Pixel abzubilden. Das ist heutzutage ausschließlich mit 3D-Elektronen-Mikroskopie (3D-EM) möglich.

Im Verlauf des letzten Jahrzehnts ist 3D-EM immer zuverlässiger geworden. Heute ist es möglich, Volumina von bis zu einem Kubikmillimeter aufzunehmen, die das gesamte Gehirn kleiner Modellorganismen, wie z.B. der Fruchtfliege *Drosophila melanogaster*, enthalten. Diese Volumina haben eine Größe von mehreren zehn oder gar hundert Teravoxeln und können nicht mehr vollständig manuell analysiert werden. Alle Aspekte der Konnektom-Rekonstruktion erfordern den Einsatz effizienter computerbasierter Methoden und Werkzeuge: (1) Die Rekonstruktion verzerrungs- und störungsfreier Volumina aus EM-Bildserien, (2) die präzise und automatische Rekonstruktion von Neuronen und Synapsen, und (3) benutzerfreundliche und effizienter Lösungen für die Visualisierung und manuelle Korrektur dieser Resultate. In dieser Dissertation beschreibe ich computerbasierte Methoden, die ich entwickelt habe, um bislang ungelöste Probleme aus allen der drei aufgezählten Bereiche zu adressieren.

In Kapitel 2 präsentiere ich eine neue Methode, planare und nicht-planare axiale Verzerrungen zu entfernen, sowie unsortierte Bildserien zu sortieren. Diese Methode war von entscheidender Bedeutung für die erfolgreiche 3D-EM-Aufnahme des ersten vollständigen Gehirns einer erwachsenen *Drosophila melanogaster*.

Maschinelles Lernen, insbesondere „Deep Learning“, sowie die Verfügbarkeit öffentlicher Trainings- und Test-Datensätze haben die automatische Rekonstruktion von Neuronen und Synapsen in 3D-EM bemerkenswert verbessert. In Kapitel 3 beschreibe ich eine neue Architektur für künstliche neuronale Netzwerke, die aus nicht-isotropen 3D-EM-Daten quasi-isotrope Rekonstruktionen von Nervenzellen generiert. Ziel ist es, eine qualitativ hochwertige Übersegmentierung aus großen Fragmenten zu erzeugen, um die manuelle Korrektur zu beschleunigen.

In Kapitel 4 beschreibe ich Programmierbibliotheken und Anwendungen, die ich für die Verarbeitung, Visualisierung und Analyse großer 3D-EM Daten und Konnektomrekonstruktionen entwickelt habe. Mit Hilfe dieser Anwendungen haben wir den derzeit größten existierenden Trainings- und Testdatensatz für die Rekonstruktion von Konnektomen aus nicht-isotroper 3D-EM generiert. Besondere Beachtung widme ich meinem flexiblen interaktiven Visualisierungs- und

Bearbeitungswerkzeug Paintera, das ich unter Zuhilfenahme dieser Bibliotheken und Werkzeuge entwickelt habe.

My doctoral studies in computer science and this dissertation would not have been possible without the help and generous support that I have received throughout my entire PhD. First, I would like to thank Prof. Fred Hamprecht of Heidelberg University for supervising my research which I was able to conduct remotely at the Janelia Research Campus. Previous work in his research group Image Analysis and Learning — including my Master's thesis — sparked my interest in bioinformatics and machine learning and I continue to stay in contact with the great people that I have met there.

I would like to thank my group leader, Dr. Stephan Saalfeld, for giving me the opportunity to work and conduct my PhD research in his lab at Janelia Research Campus, one of the best and exciting places in the world for fundamental neuroscience research. He supported me with scientific guidance and advice as well as direct contributions to my research projects. I would like to thank the entire lab for fruitful and inspiring discussions about algorithmic design and software and many enjoyable moments that we had in the office and outside Janelia as co-workers and friends: Dr. John Bogovic has shared office space with me for more than five years. During this time, he frequently helped me with research questions and software development. Igor Pisarev contributed significantly to the lab's software ecosystem and had great impact on the usability of Paintera. Larissa Heinrich shared her neural network code base and expertise to help with the design and development of my quasi-isotropic neural network architecture. In ongoing work, Chris Patrick uses Paintera and demonstrates its 3D-EM proof-reading capabilities. Neil Thistlethwaite contributed to the Paintera Conversion Helper and related software libraries during his summer internship in the Saalfeld lab. Constantin Pape shared ideas and neuron reconstruction expertise while he was at Janelia as a visiting scientist. Our lab coordinators Crystal Di Pietro, Stephanie Young, and Shelby Morris minimize the bureaucratic burden and ensure that Janelia researchers can focus on their scientific work.

Outside the Saalfeld lab, I would like like to thank Dr. Tobias Pietzsch for his work on ImgLib2 and BigDataViewer, the software stack that much of my research and software builds upon; Dr. Jan Funke for the multi-dimensional image analysis and machine-learning frameworks Gunpowder and Daisy and for sharing his visualizations of neural network architectures; Vanessa Leite for contributions to Paintera; Arlo Sheridan, Dr. Scott Lauritzen, and Ala Haddad for creating and curating dense neuron ground truth annotations; Dr. Robert Walecki and Dr. Thorsten Beier for general discussion of machine learning research and software development, their ongoing friendship, and a shared sense of humor.

I would like to thank the Howard Hughes Medical Institute and Janelia Research Campus for funding my research through the Janelia Graduate Fellowship.

I would like to thank the proof-readers of this dissertation, Stephan Saalfeld, John Bogovic and Larissa Heinrich for their helpful comments and suggestions.

Support and help from family and friends have helped me cope with the ups and downs of PhD life. I would like to thank my wife Jin Yu for her love and unconditional support, in particular in times of discouraging research or

demanding schedules. I would like to thank my parents Ursula and Paul, my brother Andreas, and my sister Katharina for staying close and visiting many times despite the ocean between us. I would like to thank my friends on either side of the Atlantic Ocean who created welcome opportunities of distraction to occupy my mind with non-research related activities.

CONTENTS

1 INTRODUCTION 17
1.1 Connectomics . . . 17
1.1.1 Electron Microscopy . . . 18
1.1.2 Neuron Reconstruction . . . 19
1.2 Contribution . . . 21
1.2.1 Publications . . . 21
1.2.2 Software . . . 21
2 ARTIFACT CORRECTION 23
2.1 Introduction . . . 23
2.1.1 Serial Section Transmission Electron Microscopy . . . 24
2.1.2 Focused Ion Beam Scanning Electron Microscopy . . . 24
2.1.3 Contribution . . . 25
2.2 Related Work . . . 25
2.3 Method . . . 26
2.3.1 Similarity Measure . . . 26
2.3.1.1 Pearson Product-Moment Correlation Coefficient 27
2.3.1.2 Best Block Matching Coefficient . . . 27
2.3.1.3 Inlier Ratio . . . 27
2.3.2 Section Order Correction . . . 28
2.3.3 Simultaneous Section Spacing and Order Correction . . . 29
2.3.4 Non-Planar Axial Distortion Correction . . . 31
2.4 Experiments . . . 31
2.4.1 Section Order Correction . . . 32
2.4.2 Spacing Correction . . . 33
2.4.2.1 Section Spacing Correction . . . 34
2.4.2.2 Comparison With Sporring . . . 35
2.4.2.3 Non-Planar Distortion Correction . . . 39
2.5 Discussion . . . 41
3 NEURON RECONSTRUCTION 43
3.1 Automatic Reconstruction . . . 44
3.2 Supervised Machine Learning . . . 47
3.2.1 Artificial Neural Networks . . . 47
3.2.1.1 Fully Connected . . . 49
3.2.1.2 Pooling . . . 49
3.2.1.3 Convolutional . . . 51
3.2.1.4 Transposed Convolutions . . . 52
3.2.1.5 Skip Connections . . . 53
3.3 Related Work . . . 53
3.4 Quasi-Isotropic Network Architecture . . . 55
3.4.1 Ground Truth . . . 59
3.4.1.1 Glial Cells . . . 60
3.4.1.2 Label Interpolation . . . 60

3.4.1.3 Limitations of Distance Transform Interpolation . 61
3.4.2 Experiments . . . 63
3.4.2.1 Augmentations . . . 65
3.4.2.2 Prediction & Reconstruction . . . 66
3.4.2.3 Results . . . 68
3.4.3 Discussion . . . 80
4 SOFTWARE 85
4.1 Paintera . . . 86
4.1.1 Related Work . . . 87
4.1.2 Architecture & Design . . . 91
4.1.2.1 Preference Pane . . . 93
4.1.2.2 Composition Modes . . . 95
4.1.3 Paintera Project . . . 95
4.1.4 Installation & Synopsis . . . 96
4.1.5 Supported Data . . . 98
4.1.5.1 Open Dataset Context Menu . . . 98
4.1.5.2 Paintera Datasets . . . 99
4.1.5.3 Raw Data . . . 100
4.1.5.4 Label Data . . . 101
4.1.5.5 Conversion Helper . . . 109
4.1.6 Controls & Shortcuts . . . 109
4.1.7 Extensions . . . 111
4.1.7.1 Source State . . . 112
4.1.7.2 Context Menu Entry . . . 113
4.1.7.3 Serialization . . . 115
4.1.8 Interactive Agglomeration . . . 119
4.1.8.1 Paintera Interactive Solver Server . . . 120
4.1.8.2 Paintera Interactive Solver Client . . . 122
4.2 EQIP . . . 123
4.2.1 Architecture . . . 124
4.2.2 Installation & Use . . . 125
4.2.2.1 Creation of Quasi-Isotropic Experiments and Setups 125
4.2.2.2 Monitoring Experiments . . . 128
4.3 imglyb . . . 130
4.3.1 Python-Java Bridges . . . 132
4.3.2 Architecture & Design . . . 133
4.3.3 Installation . . . 134
4.3.4 Usage . . . 134
4.3.5 Discussion & Future Work . . . 138
4.4 jgo . . . 139
4.4.1 Usage . . . 140
4.4.1.1 Configuration . . . 141
4.4.1.2 Examples . . . 142
4.4.2 Discussion . . . 143
Appendices 157

A ARTIFACT CORRECTION 159
A.1 Parameters . . . 159
B RECONSTRUCTION 165
B.1 Prediction and Reconstruciton Examples . . . 165
B.2 Performance Evaluation of All Parameter Sets . . . 170
C SOFTWARE 207
C.1 Paintera . . . 207
C.1.1 Data Conversion . . . 207
C.1.2 Extensions — Complete Example . . . 211

Figure 1 "Wave"-like distortion correction . . . 24
Figure 2 Similarity-based distortion correction . . . 28
Figure 3 Similarity matrices . . . 32
Figure 4 Section order correction: ssTEM-b . . . 33
Figure 5 z-position correction: FIB-SEM-a . . . 33
Figure 6 z-position correction: ssTEM-a . . . 34
Figure 7 Experiment with removed sections: finite difference gradient 36
Figure 8 Experiment with removed sections: estimated coordinates 36
Figure 9 Optimizing scale . . . 37
Figure 10 Comparison of performance with removed sections . . . 38
Figure 11 Example of artificial staining artifacts. . . . 38
Figure 12 Comparison of performance with stained sections . . . 39
Figure 13 Comparison of estimated transformation and ground truth 40
Figure 14 Automated neuron reconstruction workflow . . . 45
Figure 15 Pairwise affinities between neighboring voxels virtually increase resolution . . . 46
Figure 16 Neural network with two hidden layers . . . 49
Figure 17 2D Convolution . . . 50
Figure 18 2D Convolution . . . 52
Figure 19 2D Transposed Convolution . . . 53
Figure 20 3D U-net architectures for synaptic cleft predictions . . . 56
Figure 21 Quasi-isotropic network architecture . . . 57
Figure 22 Quasi-isotropic network architecture . . . 59
Figure 23 Distance transform interpolation . . . 61
Figure 24 Failure modes of distance transform interpolation . . . 62
Figure 25 Distance-transform based interpolation of aligned objects 63
Figure 26 Affinity predictions . . . 67
Figure 27 Prediction and reconstruction for sample 1 . . . 69
Figure 28 Glia prediction histograms . . . 71
Figure 29 Precision-recall curves for glia predictions . . . 71
Figure 30 Precision-recall curves for glia predictions . . . 72
Figure 31 Examples of affinity predictions on sample 1 . . . 73
Figure 32 Neuron reconstrution for sample A with 3D rendering of randomly selected neurons . . . 74
Figure 33 Quasi-isotropic neuron reconstruction examples . . . 75
Figure 34 Cremi metrics on 25% of the data . . . 78
Figure 35 Cremi metrics on 100% of the data . . . 79
Figure 36 Cremi metrics on 25% of the data . . . 79
Figure 37 Cremi metrics on 100% of the data . . . 80
Figure 38 Evaluation of merge threshold t_m . . . 81
Figure 39 Evaluation of glia threshold t_g . . . 82

Figure 40 Paintera proof-reading screenshot . . . 86
Figure 41 Javascript Memory . . . 90
Figure 42 Paintera layout . . . 93
Figure 43 Composition modes . . . 95
Figure 44 Paintera project . . . 97
Figure 45 Paintera latest development version . . . 98
Figure 46 Dataset dialogs for N5-like backends . . . 99
Figure 47 Raw data with various converter settings . . . 100
Figure 48 Converter pane for raw data . . . 101
Figure 49 Parameterized golden angle color assignment . . . 102
Figure 50 Triangle meshes for a neuron at multiple mipmap levels. 103
Figure 51 Automatic update of meshes after merge . . . 104
Figure 52 Mesh updates after painting . . . 106
Figure 53 3D polygon meshes from winner-takes-all downsampling and label multisets . . . 108
Figure 54 Connect to a pias instance . . . 123
Figure 55 Providing merge and split examples to a pias instance . . 124
Figure 56 Snapshot of quasi-isotropic network after 30751 iterations of training . . . 129
Figure 57 Visualization of ARGB NumPy array through imglyb . . 137
Figure 58 Paintera visualizes NumPy array . . . 139
Figure 59 Prediction and reconstruction for sample A . . . 165
Figure 60 Prediction and reconstruction for sample B . . . 166
Figure 61 Prediction and reconstruction for sample C . . . 167
Figure 62 Prediction and reconstruction for sample 0 . . . 168
Figure 63 Prediction and reconstruction for sample 2 . . . 169

Table 1	Sizes of structures in nervous system tissue.	18
Table 2	Summary of variables and parameters introduced in equations (13) to (16)	30
Table 3	Quasi-isotropic network settings	62
Table 4	Adam optimizer parameters	65
Table 5	Performance evaluation configurations	75
Table 6	Cremi metrics on 25% of the data	76
Table 7	Cremi metrics on 100% of the data	77
Table 8	Cremi metrics on 25% of the data	77
Table 9	Cremi metrics on 100% of the data	78
Table 10	Alignment of cross-sectional views and 3D scene in a two-by-two grid.	92
Table 11	Global navigation controls and shortcuts	109
Table 12	Raw-data specific controls and shortcuts	110
Table 13	Label-data specific controls and shortcuts	111
Table 14	Parameters	160
Table 15	VOI_s performance on 25% of the data	170
Table 16	VOI_m performance on 25% of the data	174
Table 17	RAND performance on 25% of the data	179
Table 18	CREMI score performance on 25% of the data	183
Table 19	VOI_s performance on 100% of the data	188
Table 20	VOI_m performance on 100% of the data	193
Table 21	RAND performance on 100% of the data	197
Table 22	CREMI score performance on 100% of the data	202

1

INTRODUCTION

Understanding the brain or, more generally, nervous systems is one of the biggest and most fascinating challenges that scientists face today. In the quest of understanding the brain, the multi-disciplinary field of neuroscience utilizes tools from many other sciences, including but not restricted to molecular biology, cell biology, mathematical modeling, physics, and many others (Ayd 2000, p. 688): Electrodes measure electrical activity within the brain (electrophysiology; Buzsaki et al. 1988); fluorescent molecules track action potentials through calcium concentrations at axon terminals (calcium imaging, T.-W. Chen et al. 2013); expansion microscopy (F. Chen, Tillberg, and Boyden 2015) surpasses the diffraction limit of light by expanding the specimen without distorting the anatomy; optogenetics (Boyden et al. 2005) controls memory (X. Liu et al. 2012; Ramirez et al. 2013) and behavior (Lin et al. 2011) with light; electron microscopes image neural tissue at synapse resolution (G. Knott et al. 2008b; Zhang et al. 2016; Eberle and Zeidler 2018) for the study of circuit connectivity in nervous systems(Sporns, Tononi, and Kötter 2005). Like in many other scientific fields, with throughput of data acquisition ever increasing, computational methods have become an essential tool for modern-day neuroscientists for curating and extracting information and scientific results from the acquired data.

1.1 CONNECTOMICS

Nervous systems are unique among organ systems (Morgan and Jeff W Lichtman 2013): A much higher diversity of cell types can be found in nervous systems, neurons with complicated geometries connect to many cellular partners through synapses to form diverse directional circuits, and the structure itself is not only formed by genetic instruction but also by the personalized experiences of each individual. Despite recent efforts, the understanding of the relation between this complex structure and function remains poor at best (Jeff W Lichtman and Winfried Denk 2011). *Connectomics* (Jeff W Lichtman and Sanes 2008) studies the *connectome* (synaptic connectivity of neurons) of a nervous systems with the goal of understanding the complex structural features of nervous systems and their relation to function. In analogy to electric circuits, this is sometimes called a "wiring diagram", albeit it currently only considers connections (while wiring diagrams consider a variety of things, *e.g.* resistor, capacitor, etc). One important and ongoing task of connectomics is the reconstruction of connectomes from micrographs. This encompasses many steps, from preparation of the specimen, over imaging and assembly of the micrograph, to the actual reconstruction of the connectome from the micrograph.

Neuronal anatomy and the idea that nervous systems consist of many neurons that are connected were first formulated in the late 19th century (Cajal 1899)

Table 1: Sizes of structures in nervous system tissue.

Structure	Size
Axons & Dendrites	$>$ 100nm
Synapses	$>$ 200nm
Membranes	$\approx$ 20nm
Synaptic Vesicles	$\approx$ 20nm
Synaptic Cleft	$\approx$ 20nm
Microtubules	$\approx$ 20nm
Gap Junctions	$\approx$ 15nm up to $<$ 3nm

but the study of neural circuit synaptic connectivity has become possible only with the availability of microscopic imaging at synaptic resolution. Starting in the 1970s, decades before the term connectome was introduced, (White et al. 1986) reconstructed the connectome of the nematode *C. elegans* from serial section electron micrographs. Even though the hermaphrodite organism has only 302 neurons, completion of the reconstruction took 14 years. The advent of high-throughput serial section electron microscopy (*EM*) in neuroscience (K. L. Briggman and D. D. Bock 2012) has made it possible to image much larger and more complex nervous systems in their entirety (Zheng et al. 2018).

The connectome is a simplified representation of a nervous system that disregards important features such as physical and chemical properties of neurons, temporal dynamics, or enzymatic processes, which are all expected to play an important role in nervous systems; advocates had to defend connectomics against the criticism of fruitlessness (Morgan and Jeff W Lichtman 2013). I believe that there is a lot of value in understanding the structure of a nervous systems: While function cannot be deducted solely from a connectome, it can certainly be constrained and tested (Seung 2009). Instead of dismissing connectomics as a futile approach at being the sole explanation of function of nervous systems, critiques would be well advised to accept it as another useful tool in the box.

In the following, I will show that modern connectomics is impossible without electron microscopy and that it is important to correct for as many imaging artifacts as possible.

1.1.1 *Electron Microscopy*

Microscopy reveals small structures that cannot be seen with the naked eye and is one of the most important tools of neuroscience. Without microscopy, it would be impossible to understand organisms at cellular and sub-cellular level. The resolution of any microscopy modality is ultimately limited by the diffraction limit (Abbe 1873) which depends on the wavelength of the electromagnetic radiation or the speed of the particle stream used for imaging. The resolution of light microscopy in the order of 100nm is too coarse to resolve relevant structures (table 1). Instead, neuroscientists resort to EM to overcome the diffraction limit of light. Modern, high-throughput EM produces vast amounts

of data at nanometer resolution. EM is used in *transmission* (*TEM*, section 2.1.1) and *scanning* (*SEM*, section 2.1.2) modes (K. L. Briggman and D. D. Bock 2012). Volumetric data is acquired as a series of two-dimensional scans (of mosaics) of sections or block-faces of the specimen, which are then assembled into a three-dimensional volume by serial section alignment (Saalfeld, Cardona, et al. 2010; Saalfeld, Fetter, et al. 2012; Wang, Gosno, and Y.-S. Li 2015). TEM and SEM differ in resolution, throughput, and contrast (K. L. Briggman and D. D. Bock 2012). During the highly precise and sensitive sample preparation and imaging process, various artifacts can be introduced, *e.g.* section folds and tears, staining artifacts, missing sections, or axial distortions.

One common artifact shared between both the TEM and SEM modalities are non-planar axial distortion along the axis orthogonal to the imaging plane: The thickness of TEM sections is not constant, and the abrasion or milling of the block-face for SEM can be variable and may not be planar across the field-of-view. Additionally, TEM sections series can be assembled in the wrong order. While the imaging quality of modern EM is excellent and error- and artifact rates are low, even a moderate presence of these artifacts can have a dramatic effect on downstream analysis, such as the reconstruction of the connectome, and thus correction of axial artifacts is addressed in chapter 2. For example, a small number of the sections in the TEM section series of a female adult drosophila brain (Zheng et al. 2018) were out of order, which had not been identified via visual inspection; instead, the method I will describe in chapter 2 was able to identify and correct for this setion misordering.

1.1.2 *Neuron Reconstruction*

Ultimately, a complete connectome requires that all neurons and their connectivity through synapses are identified. In a typical manual workflow, expert annotators trace individual neurons through volumetric data, one neuron at a time. This is a laborious task and experts spend thousands of hours to extract the connectome even for moderately sized data sets. With the throughput of electron microscopy ever increasing, analysis remains the bottleneck of connectomics (Helmstaedter 2013; Titze and Genoud 2016). One of the biggest criticisms of connectomics is that, while methods and tools have greatly improved since the first reconstruction of the *C. elegans* connectome, connectomics remains an "industrial effort" (Morgan and Jeff W Lichtman 2013): the reconstruction of the connectome of a single cortical column is estimated to require more than 10,000 years of PhD work[1,2] (Kevin L Briggman and Winfried Denk 2006). It is thus vital for the success of connectomics to automate the reconstruction process as much as possible. Typically, machine learning models learn to predict neuron boundary maps from densely annotated ground truth data. Super-voxels, or *fragments*, are then extracted from the predicted boundary maps and agglomerated

[1] https://www.economist.com/science-and-technology/2009/04/08/wired

[2] I confirmed that all web links in this dissertation are active and refer to the intended targets on June 6, 2019.

into segments. Finally, human experts proof-read the reconstruction and correct for two types of error:

UNDER-SEGMENTATION or false merge errors occur when voxels are incorrectly grouped into the same fragment or segment.

OVER-SEGMENTATION or false split errors mean that a set of voxels is incorrectly partitioned into two disjoint segments.

In general, under-segmentation can have a more catastrophic effect on the overall reconstruction result and is harder to correct for, in particular if the under-segmentation occurs at the fragment level: While over-segmentation can be corrected by simple edge contraction in the neighborhood graph of the segmentation, under-segmentation correction may require voxel-wise manipulation of fragments.

Increasingly complex machine learning models, in particular deep learning, have benefited from an unprecedented wealth of publicly available ground truth annotations and have had a tremendous impact on the reconstruction of neurons and synapses from 3D-EM. Recent advances in network architecture design have been adopted widely in neuron reconsturction: (1) In order to minimize costly under-segmentation errors, pairwise *affinities* between neighboring voxels were introduced to virtually increase the resolution of the prediction (Turaga, Kevin L Briggman, et al. 2009): Object boundaries are defined by low affinities between voxels instead of on the voxel grid. (2) Current state-of-the performance is achieved by three-dimensional variants of the popular U-Net architecture (Çiçek et al. 2016) that consider context along each spatial dimension.

Contrary to isotropic SEM, fragment extraction typically does not consider the 3D structure of the data in an anisotropic TEM setting. Instead, fragments are extracted within the 2D section planes (Funke, Tschopp, et al. 2018) and are only merged along the low-resolution z-axis during agglomeration. As a result, agglomeration models and proof-readers have to handle a much larger number of fragments.

I will present a novel artificial neural network architecture that predicts neuron boundaries at quasi-isotropic resolution from non-isotropic 3D-EM in chapter 3 to facilitate extraction of 3D fragments. The goal is to create a high-quality over-segmentation with large three-dimensional fragments for faster manual proof-reading.

Proof-reading and ground truth annotation is a challenging task for experts and user-friendly software and tools are necessary to keep up with the growing demand for dense ground truth annotations for the training of powerful machine learning models as well as for proof-reading and correcting automatic predictions. During my PhD I developed and contributed to many software libraries and tools to support the processing, visualization, and analysis of large 3D-EM data and connectome reconstructions. This software was used to generate the largest currently existing training and test data for connectome reconstruction from non-isotropic 3D-EM.[3]

[3] https://cremi.org

1.2 CONTRIBUTION

My contribution to the field of connectomics with this dissertation is three-fold:

1. Correction of axial distortion artifacts in 3D-EM (chapter 2),
2. A novel deep learning model for the automatic reconstruction of neurons at quasi-isotropic resolution from anisotropic TEM data (chapter 3), and
3. The development of software libraries and tools to support the processing, visualization, and analysis of large 3D-EM data and connectome reconstructions (chapter 4), most prominently Paintera (section 4.1), an extensible interactive proof-reading and ground truth annotation tool for connectomics that can handle arbitrarily large data.

1.2.1 *Publications*

I published my work on axial artifact correction for 3D-EM data in the ISBI conference and the journal Bioinformatics (Hanslovsky, Bogovic, and Saalfeld 2015; Hanslovsky, Bogovic, and Saalfeld 2017) and will present the Java-Python bridge for shared memory access between NumPy and ImgLib2 data structures at the 2019 SciPy conference (Hanslovsky 2019).

1.2.2 *Software*

Good software libraries and tools are vital to reproducible and meaningful modern research. Repeated execution of an experiment with the same parameter setting should always produce the same result. Software should be accessible to all scientists. Tools should be intuitive and created in close collaboration with the scientists who use them. I am a strong supporter of open source software in research and I believe that scientifc software should be accessible to all scientists. Consequently, all software libraries and tools that I developed during my PhD are publicly available on GitHub under permissible open-source licenses:

- *Paintera*[4] (section 4.1) is a user-friendly interface for efficient ground truth generation and proof-reading arbitrarily large data with 2D cross-sectional views and adaptive 3D visualization.
- *label-utilities*[5] is a Java library for manipulation of label data, most importantly distance-transform based interpolation of sections of labeled data (section 3.4.1.2)
- *EQIP*[6] (section 4.2) is a Python library for the creation and management of reproducible deep learning experiments.

[4] https://github.com/saalfeldlab/paintera
[5] https://github.com/saalfeldlab/label-utilities
[6] https://github.com/saalfeldlab/eqip

- *imglyb*[7] (section 4.3) creates a bridge between Python and Java for shared-memory access between NumPy and ImgLib2 data structures.
- Fiji plugins for the *axial distortion correction*[8] and *section sorting*[9] (chapter 2)
- *Parallelized correction of non-planar axial distortion*[10] on distributed-memory compute clusters with Apache Spark (Zaharia et al. 2010) (chapter 2)
- *flintstone*[11] helps distributing batch jobs onto the Janelia Spark cluster.
- *Fuse*[12] extends Gunpowder for training quasi-isotropic network architectures.

Additionally, I contributed to these open-source software libraries and tools:

- *jgo*[13] (section 4.4) executes arbitrary Java applications from Maven coordinates. I contributed the Python bindings and any Java program that is available through public Maven repositories can now be distributed through conda or the Python package index.
- *BigCAT*[14] is the predecessor of Paintera and was used to to generate the largest currently existing training and test data for connectome reconstruction from non-isotropic 3D-EM[15]. Most importantly, I contributed various paint modes that are also available in Paintera, *e.g.* 2D flood-fill for splitting fragments.
- *ImgLib2*[16,17] is a general-purpose, multi-dimensional image processing library. In particular, I contributed
  - A parallel and multi-dimensional implementation of sampled functions (Felzenszwalb and Huttenlocher 2012)
  - A parallel and multi-dimensional implementation of Hessian eigenvalue extraction.
  - Multi-dimensional flood-filling (also used in Paintera).
  - Connected component analysis.
  - A shear transformation.
  - Various bug-fixes and utility methods.

  I attended the 2017[18] and 2019[19] DAIS Learnathons as a tutor and ImgLib2 specialist.

[7] https://github.com/imglib/imglyb
[8] https://github.com/saalfeldlab/z-spacing
[9] https://github.com/saalfeldlab/section-sort
[10] https://github.com/saalfeldlab/z-spacing-spark
[11] https://github.com/saalfeldlab/flintstone
[12] https://github.com/saalfeldlab/fuse
[13] https://github.com/scijava/jgo
[14] https://github.com/saalfeldlab/bigcat
[15] https://cremi.org
[16] https://github.com/imglib/imglib2
[17] https://github.com/imglib/imglib2-algorithm
[18] https://imagej.net/2017-06-18_-_DAIS_learnathon
[19] https://indico.mpi-cbg.de/event/162/

2

ARTIFACT CORRECTION

Electron microscopy is one of the most important tools for connectomics: Without the nanometer-resolution of electron micrographs, synnaptic connections cannot be reconstructed. In this chapter, I present novel algorithms for the correction of non-planar axial distortions perpendicular to the image plane of electron micrographs. The work presented in this chapter was published at the ISBI conference and in the journal Bioinformatics (Hanslovsky, Bogovic, and Saalfeld 2015; Hanslovsky, Bogovic, and Saalfeld 2017).

2.1 INTRODUCTION

Serial section microscopy has been used for over a century to reconstruct volumetric anatomy of biological samples (Born 1883). Beyond its classical application in biology, zoology, and medical research, serial sectioning in combination with electron microscopy (EM) has become the de-facto standard for the reconstruction of dense neural connectivity of animal nervous systems at synaptic resolution (K. L. Briggman and D. D. Bock 2012; Stephen M. Plaza, Louis K. Scheffer, and Chklovskii 2014; Jeff W. Lichtman, Pfister, and Shavit 2014): The axial dimension of the volume is sampled by physically removing thin sections from the embedded specimen and subsequently imaging either the block-face or the section series. A resolution of less than 10nm per pixel is necessary to separate individual neural processes and to recognize chemical synapses. Sample preparation and data acquisition at this resolution are highly sensitive procedures and, as a result, imaging noise and artifacts during acquisition can be minimized at best, but not entirely avoided. In this paper, we will focus on two major acquisition modalities for large 3D electron microscopy that are used in EM connectomics: high throughput serial section transmission EM (ssTEM; Davi D. Bock et al. 2011) and block face scanning EM with focused ion beam milling (FIB-SEM; Xu and Hess 2011; G. Knott et al. 2008a; Heymann et al. 2006). While we have developed our methods with a strong focus on these two modalities, we expect them to generalize well to other applications.

We made our work available as libraries for the ImageJ distribution Fiji and for deployment in a high performance parallel computing environment. Our sources are open and available on GitHub. [1,2,3]

[1] http://github.com/saalfeldlab/section-sort
[2] http://github.com/saalfeldlab/z-spacing
[3] http://github.com/saalfeldlab/z-spacing-spark

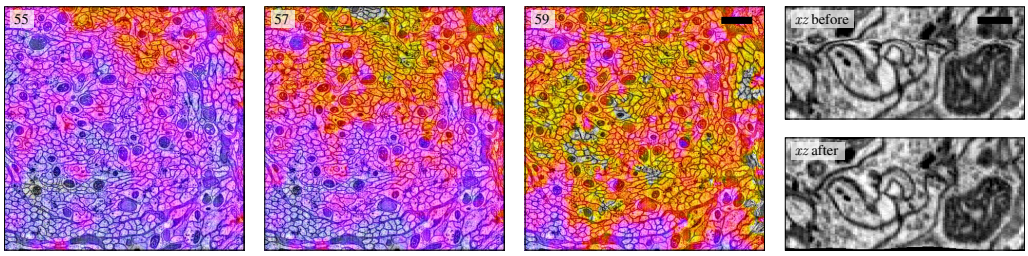


Figure 1: Left: Three FIB-SEM *xy*-section scans showing *Drosophila melanogaster* neural tissue overlaid with color-coded local *z*-spacing, serial index top left. Color overlay was chosen arbitrarily to visualize the wave-like evolution of height variance. Scale bar 1 µm. Right: Magnified crop of an *xz*-cross-section of the original (top) and corrected (bottom) series, *z*-compression by the "wave" is completely removed. Scale bar 250 nm.

2.1.1 *Serial Section Transmission Electron Microscopy*

A series of ultra-thin sections is generated by cutting the plastic embedded specimen using an ultra-microtome with a diamond knife. Section ribbons are collected on tape (K. J. Hayworth et al. 2006) or manually. Manual collection in particular bears the risk of ordering mistakes that in practice occur frequently. The nominal section thickness ranges between 30 nm and 90 nm which defines the axial resolution. Yet, discontinuous operation of the ultra-microtome and precision limits of the instrument cause variations of section thickness between and within sections. Shearing forces applied by the knife and during collection introduce deformations to individual sections. Section folds, tears, and staining artifacts further complicate the comparison of sections in the series and require that sections be aligned after imaging (Saalfeld, Fetter, et al. 2012). However, compared with block face SEM as discussed in the next paragraph, ssTEM has two major advantages: (1) sections can be post-stained which results in improved contrast of structures of interest (*e.g.* synapse T-bars), and (2) imaging is performed in transmission mode which enables high acquisition speed and significantly higher in-plane resolution at high signal to noise ratio.

2.1.2 *Focused Ion Beam Scanning Electron Microscopy*

Block face scanning EM follows a cycle of imaging the block face of a plastic-embedded specimen with a scanning electron microscope (SEM) followed by material removal until complete acquisition of the specimen. In FIB-SEM, focused ion beam (FIB) milling is used for material removal. This procedure, in practice, generates inhomogeneous *z*-spacing and non-planar block faces leading to distorted volumes (K. M. Boergens and W. Denk 2013; H. G. Jones, Mingard, and Cox 2014). These distortions exhibit a wave-like evolution of height variances throughout the acquired data and can be severe enough to seriously impede the correct reconstruction of small neural processes (figure 1). FIB-SEM has two major advantages over the previously discussed ssTEM: (1) focused ion beam milling enables significantly higher axial resolution than physical sectioning, which enables the acquisition of isotropic volumes at less than $(10\,\mathrm{nm})^3$ voxel size, and (2)

fully automatic integration of serial imaging and milling in the vacuum chamber of the microscope bears a lower risk for variances in image quality and provides better initial section alignment and correct section order.

2.1.3 *Contribution*

We developed methods for the identification and correction of ordering mistakes as well as planar and non-planar axial distortions through image-based signal analysis without the need for any further apparatus or physical measurements (section 2.3). We thoroughly assess efficacy and efficiency in virtual ground truth experiments, demonstrate their applicability to real world problems (section 2.4), and compare with state of the art (section 2.4.2.2). We published our methods as open source libraries for the ImageJ distribution Fiji and for deployment in high performance parallel computing environments using Spark (Zaharia et al. 2010).

2.2 RELATED WORK

To the best of our knowledge, both post-acquisition order correction for serial section microscopy and non-planar section thickness correction have not yet been addressed in a rigorous way. However, several methods exist for measuring or correcting section thickness or spacing. De Groot (1988) reviews four different methods for estimating section thickness, all of which require additional physical measurements, specialized apparatus, or even destructive modifications of previously acquired sections, rendering the proposed methods impractical or even impossible for certain imaging modalities, *e.g.* block face SEM. Similarly, H. G. Jones, Mingard, and Cox (2014) introduce an artifact as a fiducial mark from which section thickness can be estimated in FIB-SEM acquisitions under the assumption of planar sections. Berlanga et al. (2011) correct small volumes by evening out top and bottom surfaces that have been manually annotated by the user and transforming the whole series accordingly by a single transformation, which fails to capture varying thickness. K. M. Boergens and W. Denk (2013) reduce non-planar distortions during acquisition by using measurements of the intensity of the ion beam to control the FIB-SEM milling process. In addition, they estimate section thickness post-acquisition by adjusting z coordinates such that the peaks of auto-correlations in several xz-cross-sections have the same half-width in both dimensions.

Most related to our method for image based estimation of planar section thickness is the work by Sporring et al. (2014). They assume an isotropic signal that is sampled at less than isotropic axial resolution, with planar thickness variation. A reference similarity curve is obtained from in-section pixel intensities. The corrected spacing between adjacent sections is then determined by evaluating the inverse of the reference similarity curve at the measured pairwise similarity. We compared Sporring et al.'s method with ours and showed that our global approach has superior performance where individual sections are compromised by staining artifacts (section 2.4.2.2).

To our best knowledge, we are first to propose solely image based methods for the correction of section ordering mistakes and non-planar axial distortion. Other than existing methods for planar section thickness correction, we do not accumulate pairwise distance estimates relative to a constant reference. Instead, we jointly optimize and update the axial distortion field and an idealized observation along the axial dimension. This allows us to explicitly account for artifacts that compromise the signal in individual images, and to cope with varying properties of the reference signal along the axis of distortion.

2.3 METHOD

In the following, we describe our image-based methods for the correction of continuous and discontinuous non-planar axial distortions in serial section microscopy. Our only assumptions are — true for correct section order and spacing — monotonic decrease of pairwise similarity of sections with distance and a slow rate of change of biological tissue, *i.e.* the shape of the similarity function is locally constant (section 2.3.1). Violations of these assumptions indicate wrong section order or spacing. We will describe in detail how the coordinate space is transformed to re-establish correctness of these assumptions, and thereby correcting section order mistakes (sections 2.3.2 and 2.3.3), planar z-spacing (section 2.3.3), and non-planar z-spacing (section 2.3.4).

2.3.1 *Similarity Measure*

We define pairwise similarity $s(P_i, P_j)$ of two sections $P_i, P_j \in I$, indexed by their respective positions i, j along the z-axis within an image series that has correct section order and spacing, as a symmetric function that decreases strictly monotonically with distance $|j - i|$:

$$s : I \times I \to [0,1] \subset \mathbb{R} \tag{1}$$

$$(P_i, P_j) \to s(P_i, P_j) = f(|j - i|) \tag{2}$$

$$|j - i| < |k - l| \implies f(|j - i|) > f(|k - l|) \tag{3}$$

$$s(P_i, P_j) = s(P_j, P_i) \tag{4}$$

For a series of Z sections, all pairwise similarities are stored in a $Z \times Z$ matrix denoted by $\mathbf{S}$ such that

$$\mathbf{S}_{ij} = s(P_i, P_j). \tag{5}$$

By definition, $\mathbf{S}$ is a symmetric matrix. In practice, we use noisy surrogate measures for the inaccessible ideal s such that equation (3) may not hold for long distances. Thus, for deformation estimation, we ignore measurements for which $|j - k| > r$, for a user specified r that depends on the data set.

We implemented three similarity measures: (1) the Pearson product-moment correlation coefficient (PMCC) for aligned series, (2) the best block matching coefficient (BBMC) for approximately aligned series, and (3) the percentage of

true positive feature matches under a transformation model (inlier ratio) for unaligned data. In our experiments (section 2.4), we used PMCC and feature inlier ratio.

2.3.1.1 *Pearson Product-Moment Correlation Coefficient*

The PMCC of two statistical samples $A, B : |A| = |B| = N$ is defined as

$$\rho_{AB}(A,B) = \frac{\mathrm{cov}(A,B)}{\sqrt{\mathrm{var}(A)\mathrm{var}(B)}} \in [-1,1] \tag{6}$$

with sample co-variance

$$\mathrm{cov}(A,B) = \frac{1}{N}\sum_{i,j}\left(A_{ij}-\mu_A\right)\left(B_{ij}-\mu_B\right), \tag{7}$$

where $\mu_A = \frac{1}{N}\sum_{i,j} A_{ij}$ is the sample mean, and var(A) = cov(A,A) is the sample variance. PMCC is invariant to changes of the mean and variance of samples A and B and therefore robust against contrast and gain variations across the image series. In order to comply with equation (1), we use

$$s(A,B) = \tilde{\rho}_{AB} = \max(\rho_{AB}, 0) \in [0,1]. \tag{8}$$

2.3.1.2 *Best Block Matching Coefficient*

Similarity estimates using PMCC require the series to be perfectly aligned which, in practice, is not always guaranteed. We therefore implemented an alternative similarity measure that is robust against small local translations, the average over local best block matching coefficients (BBMC). For any rectangular region $R_i \subset P_i$, the best correspondence $R_j^* \subset P_j$ is determined by maximizing pairwise PMCC over a set of correspondence candidates $R_j \subset P_j$ of the same width and height, sampled in a small radius around the center of the region. The pairwise similarity of sections P_i and P_j,

$$\mathbf{S}_{ij} = \frac{1}{N}\sum_{R_i}\max_{R_j} \; s(R_i, R_j), \tag{9}$$

is the average of all pairwise similarities between R_i and corresponding R_j^*, where N is the total number of regions R_i within P_i.

2.3.1.3 *Inlier Ratio*

Even BBMC requires that the series is approximately aligned. In ssTEM series, however, approximate alignment is often not available, and aligning the series may be impossible because the correct order of sections has not yet been established. To recover the correct order of sections, we need a similarity measure that is independent of alignment. Using transformation invariant features, we match automatically extracted interest points across pairs of sections. We then use a variant of RANSAC in combination with a least squares local trimming estimator

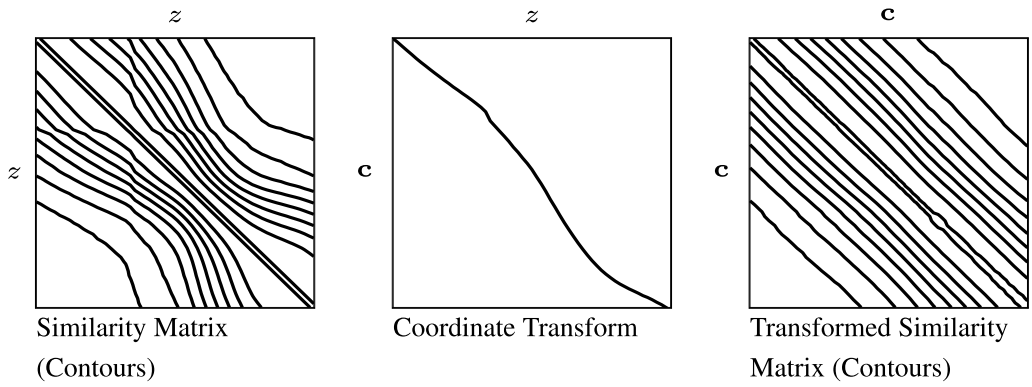


Figure 2: Warp the coordinate space such that contour-lines of the transformed similarity matrix $\mathcal{S}(\mathbf{c}_i, \mathbf{c}_j)$ are parallel to diagonal.

(Fischler and Bolles 1981; Saalfeld, Cardona, et al. 2010) to estimate a model M that transforms one set of interest points onto the other. The estimator groups all matches into inliers $\mathcal{I}$ that conform with M and outliers O that do not ($\mathcal{I} \cap O = \emptyset$). The similarity of two sections is then given by the inlier ratio

$$s(\cdot) = \frac{|\mathcal{I}|}{|\mathcal{I} \cup O|} \in [0, 1]. \tag{10}$$

For our experiments, we use SIFT (Lowe 2004). Where interest point detection and matching are part of the image alignment pipeline, (*e.g.* Saalfeld, Fetter, et al. 2012), this similarity can be extracted at virtually no cost.

2.3.2 *Section Order Correction*

Incorrect section order breaks the monotonicity assumption for similarity measures. With pairwise similarity as a proxy for distance between sections, visiting every section in the correct order is equivalent to visiting every section exactly once on the shortest path possible based on distances derived from pairwise similarity. This can be formulated as an augmented traveling salesman problem (TSP; Voigt 1831; D. L. Applegate et al. 2011). To that end, we represent the image sections $\{P_i | i = 1, \dots, Z\}$ as vertices $\mathcal{V} = \{1, \dots, Z\}$ of a fully connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with edges $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ and associated edge weights $w(a, b)\ \forall\ (a, b) \in \mathcal{E}$. Based on the intuition that sections are more similar when they are close to one another, we chose $w(a, b) = 10^5 \times \exp(|1 - \mathbf{S}_{ab}|)$ to transfer the similarities into distances.[4] With the addition of a "start" vertex $\tilde{\mathcal{V}} = \{0\}$ and zero distance edges $\tilde{\mathcal{E}} = \{(0, i), (i, 0) : \forall i \in \mathcal{V}\}$, $w(a, b) = 0\ \forall (a, b) \in \tilde{\mathcal{E}}$, establishing correct section order is equivalent to solving the TSP for the augmented graph

$$\tilde{\mathcal{G}} = (\mathcal{V} \cup \tilde{\mathcal{V}}, \mathcal{E} \cup \tilde{\mathcal{E}}). \tag{11}$$

The TSP solution will place the "start" node between the first and the last section of the sorted stack. Assuming that the first section appears before the last section

[4] While the monotonicity of this function is equivalent to the simpler $|1 - \mathbf{S}_{ab}|$, we found that stretching out similarities of nearby sections helped the TSP solver to find a correct solution.

in the initial stack, the correct order can be established by traversing the TSP solution accordingly.

2.3.3 *Simultaneous Section Spacing and Order Correction*

We observed that TSP-sorted series occasionally contain small mistakes such as flipped section pairs. Pairwise comparison alone does not capture global consistency if the similarity measure is too noisy to reliably distinguish between pairs of sections (figure 4) because similarity to any neighbor higher than first order is completely ignored. We therefore developed a method that, assuming that sections are in approximately correct order, compares shapes of complete similarity matrices to determine globally consistent order of sections and their relative spacing.

Based on the assumption of monotonically decreasing pairwise similarity and local constancy of the similarity decay, we formulate an optimization problem that simultaneously estimates a real valued position $\mathbf{c}_i$ for each section i, a scalar factor $\mathbf{m}_i$ to compensate for the influence of uncorrelated noise in individual sections to their pairwise similarity scores, and the "true" similarity $\bar{s}(\cdot)$. Correct section order can be established by sorting $\mathbf{c}$ in increasing order. We summarize all variables, parameters and measurements in table 2.

We forgo any assumptions other than monotonicity and constancy of shape in a local neighborhood. Instead, we estimate the similarity $\bar{s}_i(d)$ as a function of the distance $d = k-j$ between two sections j and k in a local neighborhood around each section i. We constrain this neighborhood by a windowing function $w_s(i,j)$ that specifies the locality of the estimate (equation (13)). These local estimates capture both changes in tissue and image properties along the z-axis. For all j within this neighborhood, the measured similarities $\delta(\mathbf{c}_j, \mathbf{c}_j + d)$ evaluated at integer distances d from the position of the section $\mathbf{c}_j$ contribute to the similarity function estimate weighted by $w_f(\cdot)$. Simultaneously, we warp the coordinate space such that all measured similarities agree with the function estimate (equation (14)). In terms of the pairwise similarity matrix that means aligning the contour lines such that they are parallel to the diagonal (*c.f.* figure 2).

Noise in individual sections decreases the pairwise similarity with all other sections in conflict to what $\bar{s}(\cdot)$ suggests and would thus distort the estimate of $\mathbf{c}$. Therefore, we estimate a scaling factor $\mathbf{m}$ for each section i to distinguish between displacement and other noise that could distort position correction (equation (15)). Using $\mathbf{m}_i$ and $\mathbf{m}_j$ to lift all pairwise similarities $\mathbf{S}_{ij}$ closer to $\bar{s}_i(\mathbf{c}_i - \mathbf{c}_j)$ will account for this effect. Sections that need displacement will not have a consistent bias towards decreased or increased similarities and remain unaffected by this "quality assessment".

The windowing function $w_r(\cdot)$ restricts the evaluation of pairwise similarities to a range r to avoid estimation based on distant sections whose similarity measures tend to be unreliable. In general, we define this window using the Heaviside step function parameterized by range r,

$$w_r(i,j) = \theta(r - |j - i|). \tag{12}$$

Table 2: Summary of variables and parameters introduced in equations (13) to (16)

Input	
$\mathbf{S}_{ij}$	Symmetric matrix containing measures of similarity for all pairs of sections indexed by i and j.

Variable	
i,j,k	Indices referencing (sub-)sections within the data.
$\mathbf{c}_i$	Location of index i in corrected coordinate space.
$\mathbf{m}_i$	Scaling factor for section i to compensate for independent artifacts.
$\mathcal{S}(\cdot)$	$\mathbf{S}$, corrected by m and warped by c: $\mathcal{S}\left(\mathbf{c}_i, \mathbf{c}_j\right) = \mathbf{m}_i \times \mathbf{m}_j \times \mathbf{S}_{ij}$
$\bar{s}_i(d)$	Local estimate of the similarity curve around i for all distances d, sampled at integer coordinates, evaluated at $d \in \mathbb{R}$.

Parameter	
$w_s(i,j)$	Windowing function that specifies the locality of similarity estimates $\bar{s}_i(d)$.
$w_r(i,j)$	Windowing function to exclude noisy similarity measures of distant sections.

Each of equations (13) to (15) contribute to a joint objective (equation (16)) that is optimized over the function estimate $\bar{s}(\cdot)$ within the support range constrained by $w_r(\cdot)$, the factors $\mathbf{m}$, and the coordinates $\mathbf{c}$:

$$\mathrm{SSE}_{\mathrm{fit}} = \sum_i \sum_j w_s(i,j) \sum_k w_r(j,k) \left(\bar{s}_i(k-j) - \mathcal{S}(\mathbf{c}_j, \mathbf{c}_j + k - j)\right)^2 \tag{13}$$

$$\mathrm{SSE}_{\mathrm{shift}} = \sum_i \sum_j w_r(i,j) \left(\bar{s}_j^{-1}\left(\mathbf{m}_i \mathbf{m}_j \mathbf{S}_{ij}\right) - \left(\mathbf{c}_i - \mathbf{c}_j\right)\right)^2 \tag{14}$$

$$\mathrm{SSE}_{\mathrm{assess}} = \sum_i \sum_j w_r(i,j) \left(\mathbf{m}_i \mathbf{m}_j \mathbf{S}_{ij} - \bar{s}_i(\mathbf{c}_j - \mathbf{c}_i)\right)^2 \tag{15}$$

$$\bar{s}^*, \mathbf{m}^*, \mathbf{c}^* = \underset{\bar{s}, \mathbf{m}, \mathbf{c}}{\arg\min}\, \alpha \mathrm{SSE}_{\mathrm{fit}} + \beta \mathrm{SSE}_{\mathrm{shift}} + \gamma \mathrm{SSE}_{\mathrm{assess}} \tag{16}$$

We find a local optimum for equation (16) by alternating least squares. In this optimization scheme, the weights α, β, and γ do not affect the arg min, and are therefore neglected. In the benign case that the series is in approximately correct order and that similarity measures capture sensible information about relative distances between sections, this local optimum is typically the correct solution. We avoid trivial solutions by meaningful regularization: all $\mathbf{m}_i$ tend towards 1, and $\mathbf{c}$ is limited by locking the first and last z-positions. If section order

is guaranteed to be correct (as in FIB-SEM), then we do not allow reordering and enforce $\mathbf{c}_{i+1} - \mathbf{c}_i > 0$ at any iteration. In addition, we enforce monotonicity of $\bar{s}$ during estimation of both $\bar{s}$ and $\mathbf{c}$. More precisely, if any scaled similarity estimate $\mathcal{S}(\mathbf{c}_i, \mathbf{c}_j)$ violates the monotonicity assumption, this measurement and all subsequent estimates $\mathcal{S}(\mathbf{c}_i, \mathbf{c}_k)$ with $|\mathbf{c}_k - \mathbf{c}_i| > |\mathbf{c}_j - \mathbf{c}_i|$ are ignored for this iteration.

2.3.4 *Non-Planar Axial Distortion Correction*

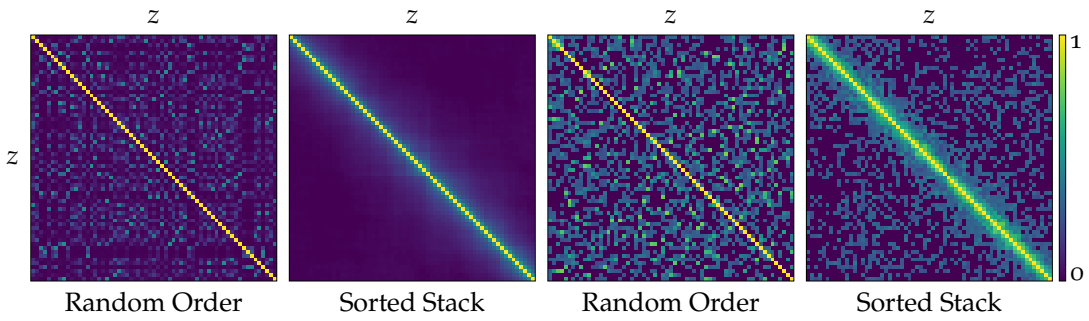
Section spacing estimation (section 2.3.3) does not require to consider complete sections but can be applied to any sub-volume defined by a local neighborhood in x and y if similarity can be estimated for pairs of sections in that sub-volume. Hence, non-planar deformation fields can be estimated by solving equation (16) for a grid of independent similarity matrices, each extracted from a local field of view. If grid locations were optimized independently, local smoothness could not be guaranteed which is particularly objectionable as similarity measures typically degrade with a smaller field of view and become increasingly susceptible to noise. Coupling terms between the optimization problems at each grid location would enforce local smoothness, but results in a single large optimization problem instead of many independent optimizations. We therefore apply a multi-scale hierarchical approach: Starting with a large field of view — typically the complete section — the spacing between sample points and the field of view around each sample point are decreased at each stage. Both parameters are freely adjustable and can result in overlapping or disjoint grid areas. Local smoothness is enforced through regularization

$$\mathrm{SSE}_{\mathrm{reg}} = \sum_i \left(\mathbf{c}_i - (\lambda \mathbf{b}_i + (1-\lambda)\bar{\mathbf{c}}_i)\right)^2 \tag{17}$$

towards the inferred coordinates $\mathbf{b}$ at the previous stage. A unique $\mathbf{b}$ for each sample point at the current stage is generated by interpolated re-sampling of the previous stage. The impact of regularization is controlled by a parameter $\lambda \in [0, 1]$ with $\bar{\mathbf{c}}$ being the result of equation (14) at each iteration of the alternating least squares solution of equation (16). All optimization problems at one stage of the hierarchy depend solely on the results of the previous stage which makes it straightforward to parallelize the solution over all grid cells. The resolution and field of view considered at each stage, the regularization parameters λ, and the range of interest (equation (12)) for pairwise similarity measurement in the z-series are exposed as adjustable parameters to the user.

2.4 EXPERIMENTS

Following the outline of section 2.3, we first describe the evaluation of section order correction, both using TSP and section spacing estimation (section 2.4.1), before we elaborate on experiments on spacing correction for planar and non-planar distortion (section 2.4.2).



(a) PMCC (b) SIFT inlier ratio

Figure 3: Similarity matrices for randomly permuted ssTEM-a series before and after section order correction. Similarities were calculated using PMCC (a) and SIFT inlier ratio (b).

2.4.1 *Section Order Correction*

We began the evaluation of section order correction with a proof of concept on a small serial section TEM data set (ssTEM-a[5]) of size $2580 \times 3244 \times 63\,\mathrm{px}^3$ and nominal voxel size $4 \times 4 \times 40\,\mathrm{nm}^3$. We perturbed the correctly ordered series using (1) a completely random permutation and (2) a permutation that randomly reassigned the position of sections within a range of ± 4 for the approaches introduced in sections 2.3.2 and 2.3.3, respectively. For (1), we evaluated both PMCC and SIFT inlier ratio as similarity measure from images scaled to 12.5% of their original size. For (2), we used PMCC only. Similarity matrices before and after section order correction are shown in figures 3 and 6 for TSP (PMCC and SIFT inlier ratio) and section spacing (PMCC and xz-cross-section), respectively. Note that, for (2), section order and z-spacing are estimated simultaneously and therefore the sorted matrix appears warped. TSP re-established the correct section order for both PMCC and SIFT inlier ratio. The run times for optimization of the TSP problems are negligible compared to the time required to extract pairwise similarities (14 ms vs. 720 ms for PMCC and 19 ms vs. 9064 ms for SIFT inlier ratios). Shorter run times for solving the TSP in the PMCC experiment indicate that, with PMCC, the problem is easier due to better similarity measures. PMCC is superior to SIFT inlier ratio as a similarity measure for well aligned series. Even for larger examples, the run time for the TSP solution remains short, *e.g.* 2070 ms for 2051 sections (data not shown). All experiments were carried out on a Dell Precision T7610 workstation using the TSP solver *concorde* (D. Applegate et al. 2006).

With this successful proof of concept at hand, we proceeded with section order correction of a longer section series (ssTEM-b[5]). We chose an unaligned series of 251 complete sections for which we manually curated the correct section order. The objective of the experiment was to re-establish correct section order from an initially unaligned series with ordering mistakes. We therefore extracted the SIFT inlier ratio matrix from the unaligned series and estimated section order via TSP. The solution included small pairwise ordering mistakes. However, these

[5] ssTEM of *Drosophila melanogaster* CNS, courtesy of D. Bock, R. Fetter, K. Khairy, E. Perlman, C. Robinson, Z. Zheng, HHMI Janelia

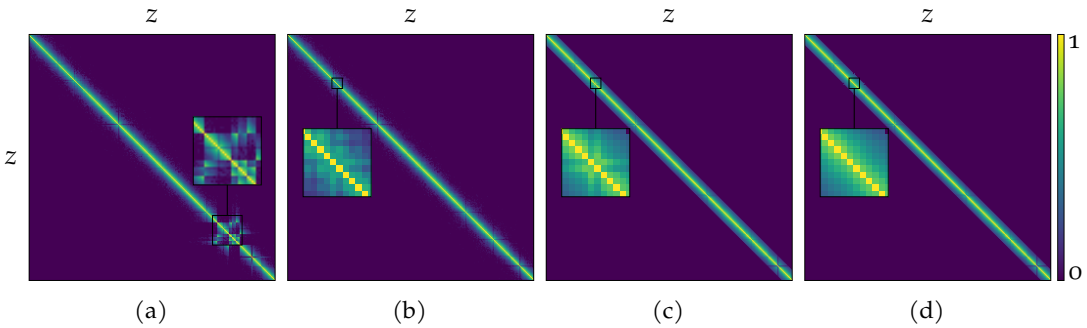


Figure 4: Section order correction for ssTEM-b. Inlier ratio matrix for original sequence (a) and after correction (b). The major disturbance (bottom right) could be resolved but two sections remain flipped (magnified view). This becomes more apparent in the PMCC matrix of the aligned series (c). Repeated TSP correction resolves this remaining issue (d).

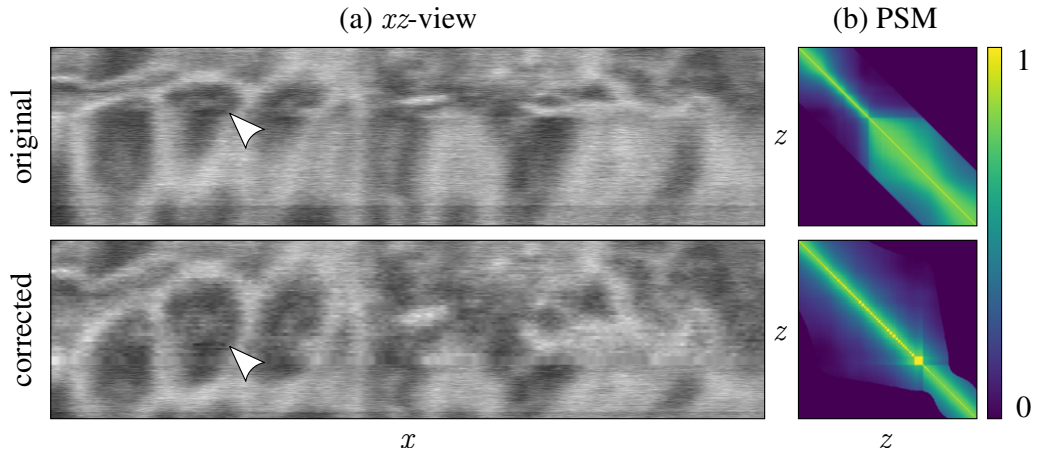


Figure 5: z-position correction experiment for FIB-SEM-a: Top/bottom show an xz-cross-section (a) and corresponding intensity-encoded pairwise similarity matrices (PSM;b) before/after z-position correction. (a) and (b) share the same coordinate frame in z. Arrows highlight areas that are visually stretched or compressed in the original acquisition and appear biologically plausible after correction.

disturbances were sufficiently local to enable elastic alignment (Saalfeld, Fetter, et al. 2012) of the corrected series and to extract a PMCC similarity matrix. We then used the TSP method to estimate order from the PMCC similarities (figure 4), decreasing the number of misplaced sections from 2 (0.80%) to zero.

2.4.2 *Spacing Correction*

Similar to the experiments for section order correction, we started with a proof of concept, followed by an extensive experiment for the evaluation of non-planar distortion correction using an artificial ground truth deformation on a real world data set.

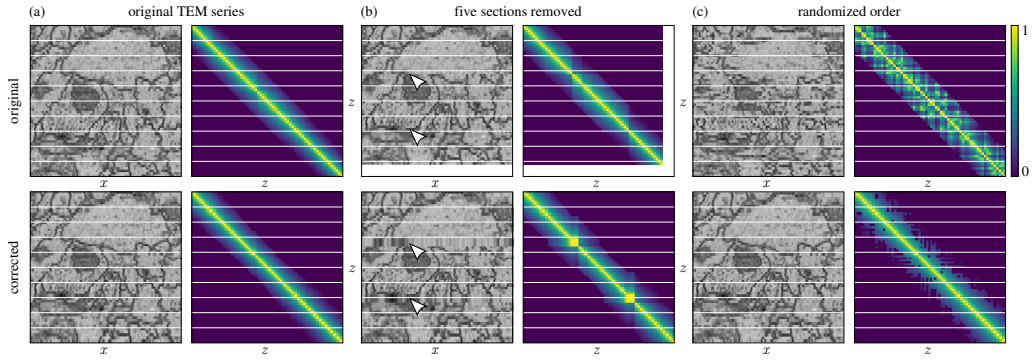


Figure 6: z-position correction experiments for ssTEM-a: original series (left), missing sections (center), and randomized order (right) with a shared coordinate frame in z, as indicated by the white grid. Top/bottom show an xz-cross-section (left sub-column) and corresponding intensity-encoded pairwise similarity matrices (right sub-column) before/after z-position correction. Arrows in the center column highlight removed sections.

2.4.2.1 *Section Spacing Correction*

For the evaluation of section spacing correction, we corrected and visually inspected distortions in two data sets: ssTEM-a and FIB-SEM-a[6] with dimensions $2048 \times 128 \times 1000\,\mathrm{px}^3$ and nominal voxel size $8 \times 8 \times 2\,\mathrm{nm}^3$. The latter is an excerpt of a larger data set with dimensions chosen such that axial distortions can be considered approximately planar.

Figure 6 shows xz-cross-sections and the according similarity matrices before and after section spacing correction for (a) the original ssTEM-a data set, (b) sections 20, 21, 22, 46, 48 removed, and (c) randomized section order. Our experiments show that for the original data set, z-spacing varies between 0.6 px and 1.6 px (24 nm and 64 nm). Section spacing correction of (b) and (c) was evaluated by comparing the estimated transformations with the result of (a) as "ground truth". The estimated transformation for (b) correctly stretches the data where sections were removed and deviates (absolute value) from the ground truth by 0.13 px (5.2 nm) on average, and not more than 0.28 px (11.2 nm). Sections removed for this experiment do not contribute to the evaluation. For the simultaneous order and spacing correction (c), we measured an absolute deviation from the ground truth of 0.044 px (1.76 nm) on average, and not more than 0.13 px (5.2 nm). All ssTEM-a section spacing correction experiments finished in 0.6 s (similarity matrix calculation) and 0.4 s (inference, 100 iterations) on a Dell Precision T7610 workstation using the default parameters of the provided Fiji plugin.

We observed stronger distortions in FIB-SEM-a as shown in figure 5 (top). Stretched/condensed regions are highlighted in an xz-cross-section and appear in the respective similarity matrix as regions with slow/fast decay of similarity. After section spacing correction (figure 5 bottom), the corrected xz-cross-section appears homogeneously sampled and similarity decay is approximately constant. The estimated section spacing varies between 0.14 px and 10.2 px, or 0.28 nm

[6] FIB-SEM of *Drosophila melanogaster* CNS, courtesy of K. Hayworth, H.Hess, C. Shan Xu, HHMI Janelia (Kenneth J. Hayworth et al. 2015)

and 20.4 nm. On the Dell Precision T7610 workstation used for this experiment, similarity matrix estimation and inference (150 iterations) took 62.3 s and 49.4 s, respectively using the default parameters of the provided Fiji plugin, with the exception of $r = 55$ (equation (12)).

2.4.2.2 *Comparison With Sporring*

We compared our method with the work by Sporring et al. (2014) which is most related to ours. Please note that Sporring et al.'s method addresses planar thickness estimation in non-isotropic section series against a constant reference, but not section series order correction or non-planar distortion correction. Therefore, this comparison covers only planar thickness correction for non-isotropic data.

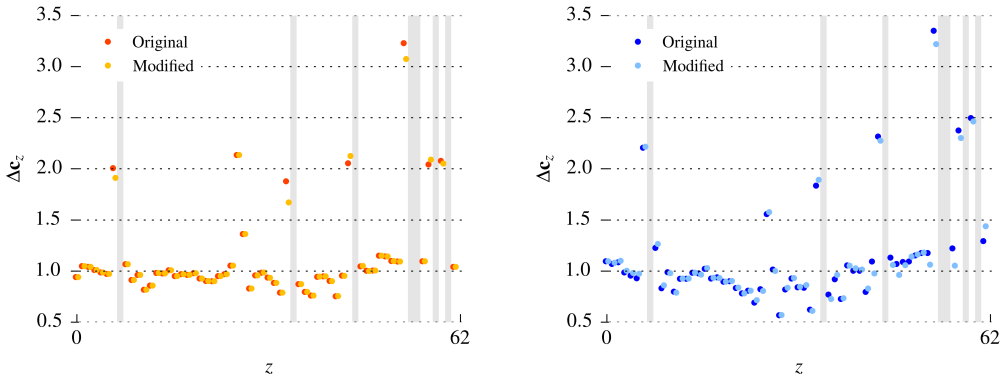
We used the Transmission Electron Microscopy (TEM) series ssTEM-a (section 2.3.2) from Hanslovsky, Bogovic, and Saalfeld (2015) for all subsequent experiments. For lack of ground truth, we considered the respective thickness estimates for each method as 'correct' solutions. These solutions are not the same for both methods as shown in figure 8. We believe that this difference can be attributed to the accumulation of false estimates by (Sporring et al. 2014) caused by imaging and preparation artifacts. However, we did not consider this in the evaluation, and, in all subsequent experiments, compared the results to each method's own reference. We used the example source code that Sporring et al. provided in their publication with a small modification: we prepended $\sigma_0 = 0$ to the measured standard deviations as a proxy for comparing a section to itself, which is necessary to estimate distances along z that are smaller than the resolution within xy. In our experiments, we introduced known modifications to the ssTEM series and then compared the predictions on the modified series with the reference to evaluate correctness and robustness of each method. Ideally, modifications to the series should be identified and the reference solution should be reproduced. In particular, we modified the series by (a) removing varying numbers of sections at random locations, and (b) adding artificial circular stains that reasonably resemble staining artifacts that we observe in real data. We repeated each randomized experiment with 25 different seeds for each set of parameters. Before we continue with the experiments, we introduce terminology and definitions. In agreement with the main text, we denote the estimated z-coordinates by $\mathbf{c}$. The reference coordinates are indicated by $\bar{\mathbf{c}}$. We denote the forward difference approximation of the gradient of the estimated coordinates $\mathbf{c}$ as

$$\Delta\mathbf{c}_i = \mathbf{c}_{i+1} - \mathbf{c}_i. \tag{18}$$

As Sporring et al.'s method generates $\Delta\mathbf{c}$ rather than $\mathbf{c}$, we derive $\mathbf{c}$ as the cumulative sum over $\Delta\mathbf{c}$.

Removed Sections

We found our own method to perform optimally when images are scaled to isotropic or sub-isotropic resolution. ssTEM-a is non-isotropic at a ratio of approximately 1:10. We therefore used a scale factor of 0.05 applied to all section images when testing our own method. For Sporring et al.'s method, we tested (a)



(a) Sporring et al. (b) Ours

Figure 7: Finite difference gradient for original series and with a total of seven removed sections as indicated by shaded areas in the plots. Where sections $\{m, \ldots, n\}$ have been removed, no gradient is calculated. The gradient of the coordinates for the original series at position $z = m - 1$ is estimated by $\mathbf{c}_{n+1} - \mathbf{c}_{m-1}$. For a correct estimate of the spacing of the removed sections, the gradient of the transformation for the modified series must be the same as the gradient for the original series at all relevant locations. For the purpose of visualization, there is a small offset between the gradients for the original and the modified series.

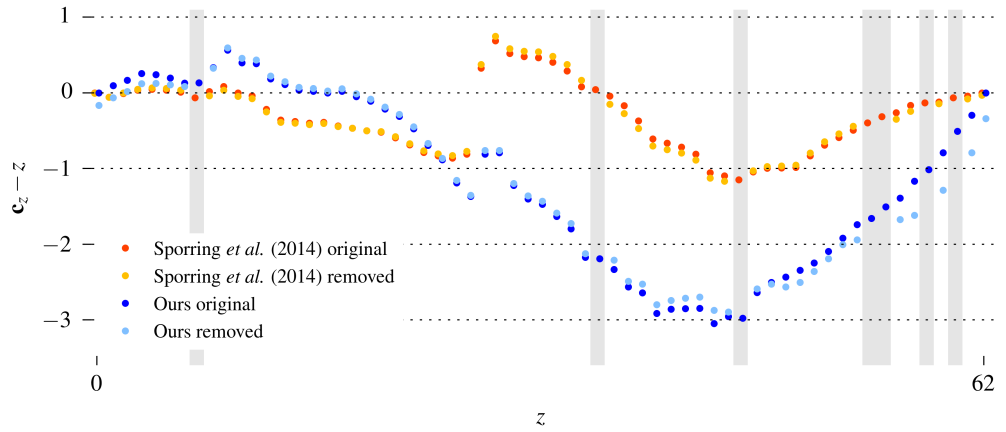


Figure 8: Estimated coordinate mapping for original series and with a total of seven removed sections as indicated by shaded areas in the plots (Sporring et al. (2014) and ours). Predicted coordinates are mapped to the corresponding reference with a linear function that minimizes the square displacement between corresponding coordinates. For the purpose of visualization, we introduced a small offset along the x-axis.

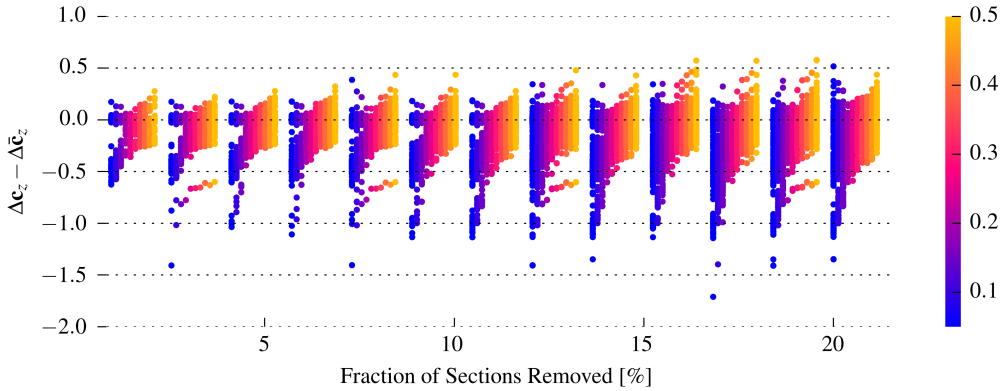


Figure 9: Optimzing scale for Sporring et al. (2014). Color encodes the scale factor of the image series in xy as indicated by the color bar.

for ten different scale factors $(0.05, 0.1, \ldots, 0.5)$. We found that Sporring et al.'s method generally performs poorly at isotropic or sub-isotropic scales. This means that, other than our method, Sporring et al.'s approach is not ideal for isotropic data or for data where the axial resolution is higher than the planar resolution (e.g. FIB-SEM). We picked the best performing scale factor 0.4 for comparison with our method (figure 9). We also used that scale factor for (b). For each experiment, we show scatter plots of the differences $\mathbf{d} = \Delta\mathbf{c} - \Delta\bar{\mathbf{c}}$ between estimated and ground truth distances between adjacent sections. In order to account for irrelevant scale differences, we mapped the references for both Sporring et al.'s method and ours onto the initial interval $[0, 62]$ and scaled $\mathbf{c}$ such that the sum of all $\mathbf{d}_i$ that were not affected by modifications is minimal. This way we guarantee maximal fairness to Sporring et al.'s method where all unmodified $\mathbf{d}_i$ can be perfectly reproduced.

For each number $n \in \{1..13\}$, we generated 25 image series with n randomly chosen sections removed from the series. A single (13) removed section(s) is equivalent to roughly 2% (21%) of all sections. Figure 8 shows an anecdotal example of coordinate mapping estimates for a series with 7 removed sections, taken from this series of experiments. It highlights the qualitative differences of both methods. In the following, scatter plots of gradient differences as shown in figures 7(a) and 7(b) form the basis for our evaluation. Figure 9 indicates that Sporring et al.'s method performs best at a scale of 0.4 in xy. Therefore, we scaled each image series by 0.4 before applying their method. For our method, we scaled each series by a factor of 0.05, and modifed the default options to use a range of $r = 6$ neighboring sections, run for 1000 iterations, and disallow re-ordering of the series.

The scatter plots of $\mathbf{d}$ shown in figure 10 indicate that–for a small number of removed sections–both methods are able to accurately reproduce the respective reference and estimate the correct spacing, including removed sections with an error of less than 0.25 px. With increasing number of sections removed, performance decreases for both methods. Sporring et al.'s method performs slightly better than ours which we attribute to it being less affected by false initial estimates of the reference similarity function and considering only pairwise differences.

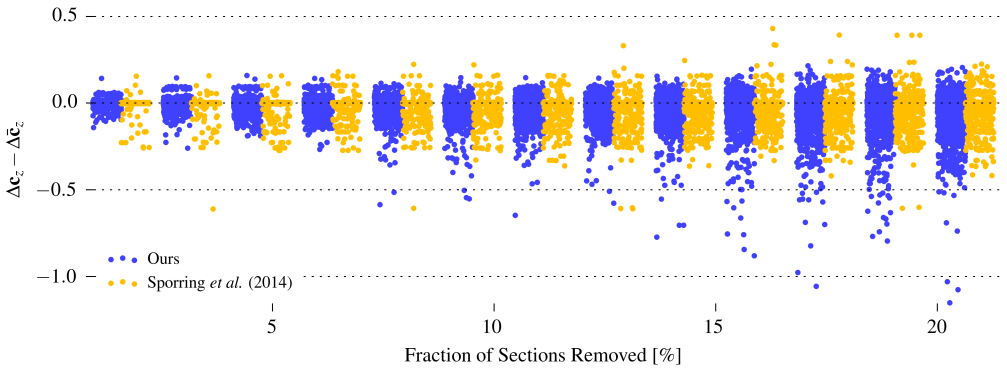


Figure 10: Comparison of performance for removed sections between Sporring et al. (2014) and ours. The spread along the x-axis is introduced for the purpose of visualization.

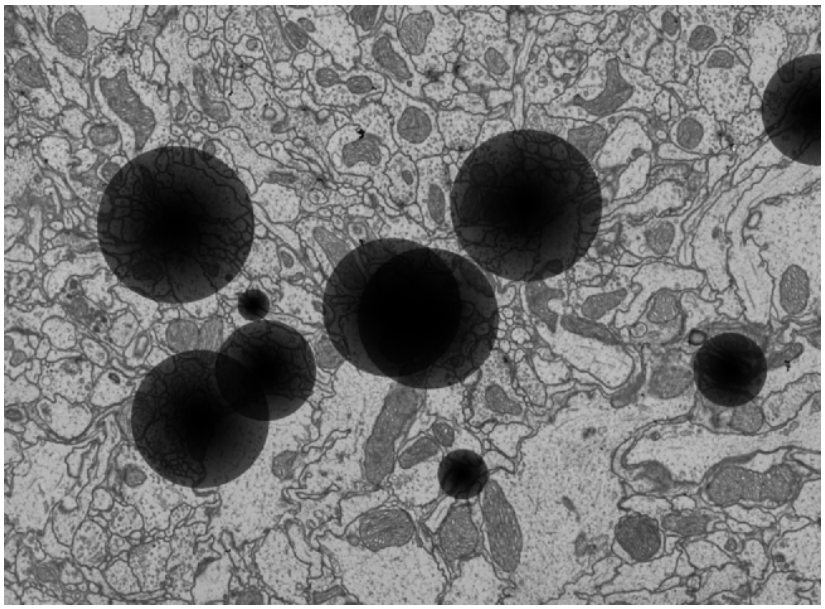


Figure 11: Example of artificial staining artifacts.

In our method, the reference similarity function is compromised by removing sections from the series which manifests as increasing deviation from the reference solution. For both methods, the error remains low, almost always below 1 px which corresponds to the average thickness of one section.

Staining Artifacts

In order to simulate EM staining artifacts, we added circular speckles to randomly selected sections, such that for any circle with radius r, and center x_0, y_0, we multiply (degrade) the intensity at each location

$$(x, y) \in \{(x, y) \in \mathbb{R}^2 | \sqrt{(x - x_0)^2 + (y - y_0)^2} \leq r\} \tag{19}$$

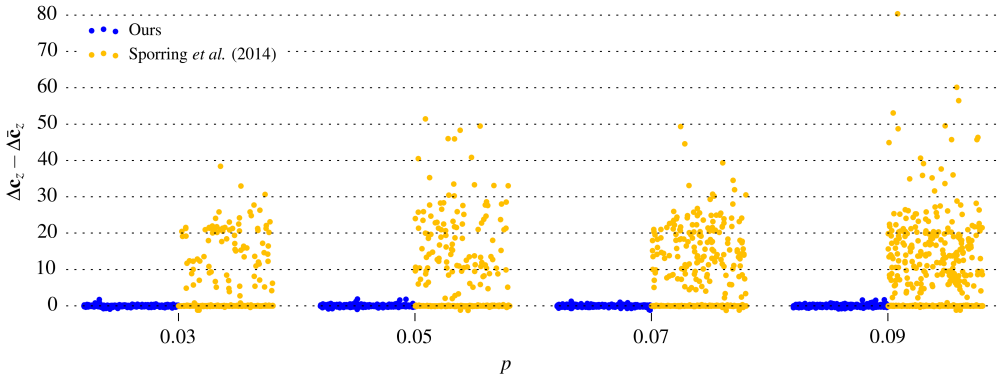


Figure 12: Comparison of performance for stained section between Sporring et al. (2014) and ours. The spread along the x-axis is introduced for the purpose of visualization.

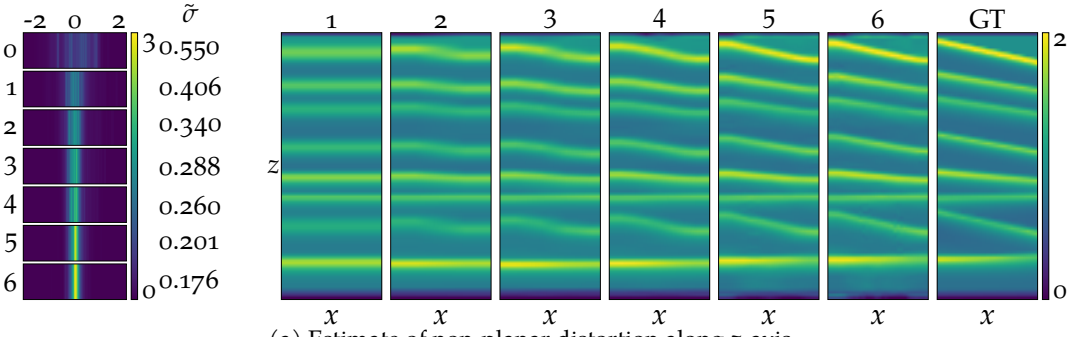
by a factor

$$f = 1 - e^{-\frac{1}{2}\frac{(x-x_0)^2+(y-y_0)^2}{r^2}}. \tag{20}$$

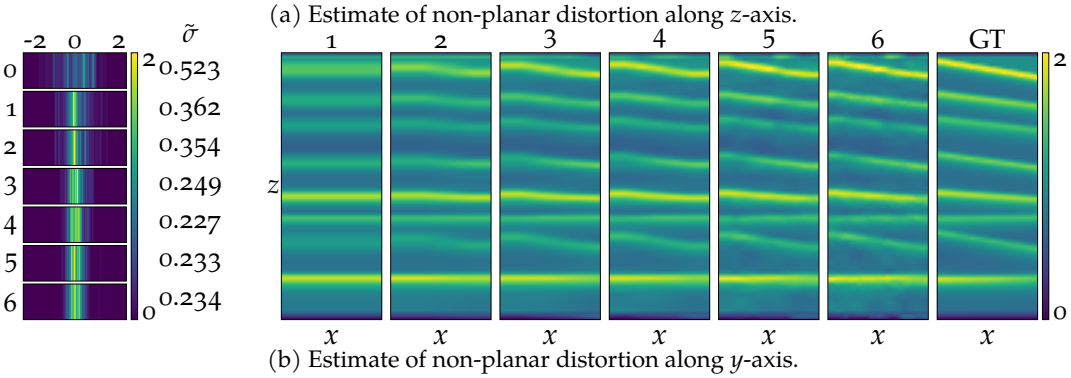
Be $p \in \{0.03, 0.05, 0.07, 0.09\}$ the staining artifact probability. We ran 25 experiments with different random seeds for each p. In each experiment, we drew any section with a probability of p. For any drawn section, we created 20 circles centered at random locations and with radii uniformly distributed over the interval $[20, 400]$ px. Each circle then stains the image with a probability of 0.5. An example image with artificial staining artifacts is shown in figure 11. Again, we chose a scale of 0.4 in xy for (Sporring et al. 2014). For our method, we picked a scaling factor of 0.05 for xy, and modifed the default options to use a range of $r = 4$ neighboring sections, run for 200 iterations, and disallow re-ordering of the series. The scatter plots of $\mathbf{d}$ in figure 12 show that Sporring et al.'s method strongly overestimates the distance between sections if at least one of them displays staining artifacts. In our method, the consideration of more than pairwise competing distance predictions and the scalar factors for compensation of uncorrelated noise address this problem effectively such that we can reliably reproduce the reference.

2.4.2.3 *Non-Planar Distortion Correction*

We evaluated the performance of non-planar deformation correction against synthetic ground truth. To that end, we applied synthetic non-planar axial distortion to a distortion free reference series, estimated the distortion with our method (section 2.3.4), and compared the estimate with the synthetic ground truth. Since distortion free volumes do not exist, we had to first correct the original image volume using the same non-planar axial distortion correction method. The resulting series, from the perspective of our method, is free of distortions. To compensate for the apparent bias in this approach, we ran our experiment not only in the original orientation but permuted the coordinate axes such that the new axial dimension falls into the unprocessed image plane.



(a) Estimate of non-planar distortion along z-axis.



(b) Estimate of non-planar distortion along y-axis.

Figure 13: Normalized histograms (left) of differences between estimated transformation and ground truth (GT) and visualization of the estimated transformation for all stages and GT for experiments on both sub-volumes (a) and (b) via xz cross-sections of the gradient (right). Histogram bins range from -2 px to 2 px, with maximum counts of 3 and 2 for (a) and (b), respectively. The gradients range from 0 px to 2 px.

The data used in this experiment is a subset of FIB-SEM-b[6] with dimensions $4000 \times 2500 \times 2100\,\text{px}^3$ and voxel resolution $8 \times 8 \times 2\,\text{nm}^3$. Initial non-planar axial distortion correction was distributed onto 60 compute nodes with 16 cores each and took 120 minutes to finish. We scaled the corrected series along the z-axis by a factor of 0.25 resulting in an isotropic volume of $4000 \times 2500 \times 525\,\text{px}^3$ from which we extracted two sub-volumes: (a) 100 complete xy-sections starting at $z = 25$, and (b) 100 xz-cross-sections of dimension $3000 \times 475\,\text{px}^2$ starting at $y = 1000$. For (b), we flipped the y- and z-axes such that the synthetic distortion could be consistently applied along the z-axis. Our synthetic distortion model is this: Randomly oriented planes superimposed with trigonometric functions act as attractors that shift the coordinates towards the attractor along the z-axis as a monotonically decreasing function of the distance to the attractor along z. This generates waves and plateaus that approximately resemble phenomena that we observed in the original volume before correction. We then applied non-planar distortion correction to the synthetically deformed series as described in section 2.3.4 (parameters listed in appendix A.1). We compared estimated and ground truth distortions at every stage of the hierarchical solution and show histograms of the pixel-wise differences (figure 13). Since we are not interested in low frequency distortion of the volume, we mapped each estimate onto the ground truth using a linear transformation that minimizes the squared difference of corresponding look-up table entries within local support defined by a Gaussian window with $\sigma = (\sigma_x, \sigma_y, \sigma_z)$.

The evolution of the estimated distortion for each of the sub-volumes is shown shown in figure 13. For intuitive visualization, the gradient is displayed. Starting at a complete field of view and a resolution of 1 px² in x and y at stage 1 (planar estimate), the field of view/resolution is decreased/increased by a factor of two in both x and y with every sub-sequential stage which allows for a more accurate estimate of the deformation. At the same time, noise in the data will have a stronger influence on smaller fields of view (*c.f.* figure 13, stage 6) and sets a limit to the resolution at which the deformation can be estimated. The histograms of differences did not improve after (a) stage 6 or (b) stage 4. We chose $\sigma = (\infty, \infty, 120\,\text{px})$ for the Gaussian window to estimate the linear transformation. The mean of differences between estimate and ground truth is approximately zero for all stages. We therefore used the standard deviation of the error $\tilde{\sigma}_i$ for stage i including the baseline $i = 0$ as a quality indicator. Smaller $\tilde{\sigma}_i$ means better estimates of the ground truth. For (a), we found $\tilde{\sigma}_0 = 0.550\,\text{px}$ and $\tilde{\sigma}_6 = 0.176\,\text{px}$, and for (b), we found $\tilde{\sigma}_0 = 0.523\,\text{px}$ and $\tilde{\sigma}_4 = 0.227\,\text{px}$. As expected, non-planar axial distortion correction considerably decreased the distortion of the series in both experiments.

2.5 DISCUSSION

We developed novel methods to address two previously unsolved problems: (1) establish the correct order of unordered section series, (2) compensate for planar and non-planar axial distortion. We demonstrated through extensive experiments that our methods work reliably and with high accuracy and efficiency on both ssTEM and FIB-SEM data. We went beyond pure proof of concept and showed that our methods are applicable to and perform well on large real world data sets.

In large ssTEM series, the combination of automatic alignment and series sorting has the potential to greatly reduce the need for manual intervention. Non-planar axial distortion correction addresses the peculiar wave-problem in FIB-SEM which, we believe, will have a strong impact on the future application of FIB-SEM for high resolution 3D reconstruction.

In this work, we made only mild assumptions about the data, *i.e.* monotonic decrease of pairwise similarity and local constancy of the shape of the similarity curve. While this means that our methods can be applied to a wide range of data, we predict that many problems would benefit from domain specific modeling. For example, explicit modeling of FIB-SEM-waves has the potential to further increase the accuracy of the estimated deformation field. We will work on these ideas in our future research.

3

NEURON RECONSTRUCTION

A synaptic connectome consists of all neurons and the synapses connectimg them. Reoconstructions from 3D-EM been achieved through manual expert annotation for small to moderately-sized data sets: In laborious efforts and thousands of hours, expert annotators traced neurons in serial section electron micrographs (Eichler et al. 2017). While computational power of computers, tools, and user interfaces have greatly improved (Saalfeld, Cardona, et al. 2009) since the reconstruction of the first connectome (White et al. 1986), which took 14 years to completion, it is clear that reconstruction of larger (parts of) nervous systems or even multiple specimens is intractable — both financially and temporally — with a purely manual approach. Instead, the power of modern computers must be utilized to automate neuron reconstruction and to minimize the human effort that is required to reconstruct connectomes.

Supervised machine learning (section 3.2), in paticular deep neural networks section 3.2.1, have driven the performance of automatic reconstruction closer to human performance than ever before (Beier et al. 2017). Neuron segmentation cannot be trained end-to-end. In a typical automatic neuron reconstraction workflow (section 3.1), deep learning models predict boundary maps from which *fragments* (super-voxels) are extracted. These fragments are then agglomerated into *segments* to complete the reconstruction (Beier et al. 2017; Funke, Tschopp, et al. 2018). Boundary map prediction and fragment extraction are usually parallelized over moderately sized sub-volumes of the data but agglomeration is a global operation. Ultimately, the number of the fragments is the limiting factor for agglomeration, in particular for complex models like multi-cuts (Beier et al. 2017). This is especially severe in anisotropic TEM because, typically, 2D — and therefore many more — fragments are generated.

I developed a novel deep learning network architecture for neuron boundary prediction at *quasi-isotropic* resolution from *anisotropic* TEM (section 3.4). Larger, three-dimensional fragments can be extracted from quasi-isotropic boundaries to reduce the number of fragments. In order to utilize existing anisotropic ground truth annotations and to avoid laborious manual annotation of data, I developed a novel scheme for the interpolation of sections of label data (section 3.4.1.2). I evaluated performance in extensive experiments (section 3.4.2) and show promising results of quasi-isotropic boundary prediction and neuron reconstruction from anisotropic TEM. Finally, I will discuss current limitations and future research directions to fully utilize the capacity of neuron reconstruction at quasi-isotropic resolution in section 3.4.3.

First, I will start with a more detailed description of a typical automatic neuron reconstruction workflow (section 3.1), followed by an overview of relevant supervised machine learning techniques, in particular artifical neural networks, in section 3.2 and related work (section 3.3).

3.1 AUTOMATIC RECONSTRUCTION

In a typical manual reconstruction workflow, annotators create sparse tree representations of neurons, also known as *skeletons*. Tracing a single neuron through section series, annotators add prominent points to the neuron tree. Sub-trees can be merged or split to combine and correct existing skeletons, and synapses can be annotated to connect individual skeletons (Saalfeld, Cardona, et al. 2009). Sparse reconstructions minimize the effort for annotators compared to dense reconstructions but valuable information about neuron anatomy is lost and ambiguity is added: Any dense segmentation could be represented by many different skeletons.

Automatic reconstruction, in contrast, typically reproduces a *dense* representation of the neurons and is a special case of the traditional *instance segmentation* task in computer vision for *3D* data: all voxels of a three-dimensional volume are clustered into distinct objects of interest, also called *segments*. Neuron reconstruction has additional challenges compared to classical 2D image segmentation:

LARGE AMOUNTS OF DATA are generated for single nervous systems, even for small model organisms: The electron micrograph of a single adult *Drosophila melanogaster* brain imaged at the resolution necessary to reconstruct all neurons and synapses consists of hundreds of teravoxels.

SAMPLE PREPARATION AND IMAGING AT THE NANOMETER SCALE make it impossible to generate artifact-free data. This includes section folds and tears, missing sections, or incomplete cell boundaries.

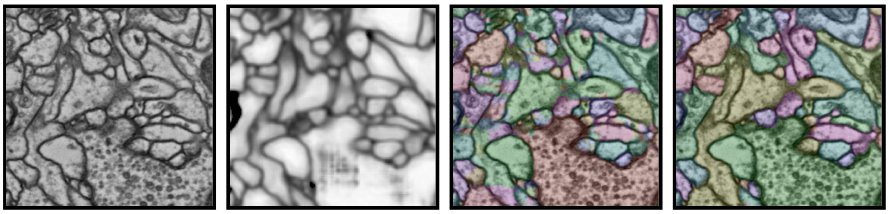
REGISTRATION OF SECTION SERIES has a huge impact on the segmentation result. Bad registration can render neuron reconstruction impossible.

NEURON SHAPES ARE NOT COMPACT and individual neurons span a substantial subspace of the entire micrograph. Decomposition of the segmentation problem into smaller, independent problems is impossible and seemingly minuscule local mistakes can have a huge impact on the global segmentation result. Note that over-segmentations errors are preferred over under-segmentations because they are easier to fix for proof-readers.

THE DATA IS DENSELY PACKED WITH VISUALLY SIMILAR NEURONS and it is impossible to distinguish between objects based on appearance features. Instead, classifiers need to detect boundaries, which can be incomplete. In the ideal case of flawless boundary predictions, the neurons could be reconstructed as connected components from the (inverse) boundary map. For real world problems, a more involved workflow is required that I will describe below.

THE ADDITIONAL SPATIAL DIMENSION increases neural network model size dramatically and also increases the risk of false negative boundary classification between two distinct objects.

SPATIAL ANISOTROPY in ssTEM data introduces another source of reconstruction error as the nominal resolution of 40nm orthogonal to the sectioning plane is rather coarse compared to small neural processes and synapses (table 1).



(a) raw data (b) boundaries (c) fragments (d) segments

Figure 14: Automated neuron reconstruction workflow. Machine learning classifiers predict boundaries (b) from raw electron micrographs (a). The boundary map is partitioned into fragments (c) that are then agglomerated into segments (d). Experts proof-read the final segmentation to ensure correctness and correct for over- and under-segmentation errors.

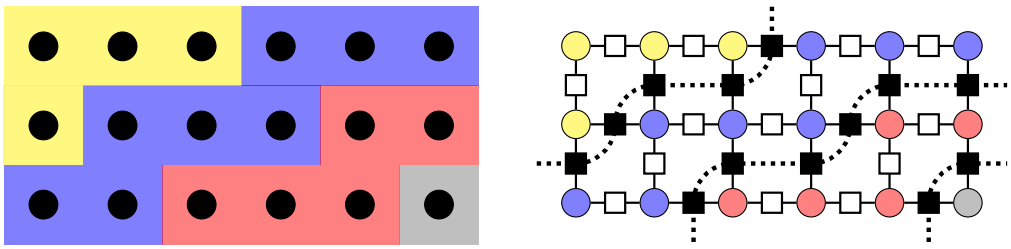
GROUND TRUTH DATA is not as readily available as in natural image processing and laborious generation requires expert knowledge. Recent developments such as the CREMI challenge[1] addressed this problem but the thirst for training data of mega-parameter artificial neural networks will probably never be satisfied.

These complicating factors have made *end-to-end* neuron reconstruction impossible to date; currently there is no single model that directly predicts a segmentation from raw data. Instead, with few exceptions, cell boundaries are used as proxies for objects. In practice, the boundary predictions are not good enough to extract objects through simple thresholding, and a multi-step workflow of advanced image processing, computer vision, and machine learning techniques is employed to reconstruct neurons as well as possible (figure 14):

1. Predict neuron *boundaries* from raw image data with a boundary classifier.
2. *Over-segment* the boundary predictions into *fragments* or *super-voxels*, meaningful clusters of voxels that (ideally) respect neuron boundaries, similar to *super-pixels* in two-dimensional image processing.
3. *Agglomerate* the fragments into segments. Clustering schemes like hierarchical clustering (Nunez-Iglesias, Kennedy, Stephen M Plaza, et al. 2014) or multi-cut (correlation clustering) (Bansal, Blum, and Chawla 2004; Bjoern Andres, Kappes, et al. 2011; Bagon and Galun 2011; Kim et al. 2011; Yarkony, Ihler, and Fowlkes 2012) are popular choices for this task. The parameters of these agglomeration can be learned. Under-segmentation errors at the fragment level (2) cannot be corrected for during agglomeration. Ideally, these segments coincide with the biological neurons that are present in the data.
4. *Proof-read* the final reconstruction to confirm correctness and manually correct for any mistakes in the automatic segmentation or agglomeration.

Boundary predictions can either be scalar voxel-wise boundary probabilities or a vector of weights that describe the local *affinity* to neighboring voxels, *i.e.* how likely

[1] https://cremi.org/



(a) (b)

Figure 15: It can be impossible to create a correct segmentation with scalar boundary predictions in a scenario where resolution is not much higher than the spatial extent of small objects and foreground objects are adjacent without room for boundary voxels in between. All three colors yellow, blue, and red represent valid objects in (a). Specifically, there are no boundary pixels (gray background) between any of these objects. The true segmentation cannot be recovered with voxel-wise boundary predictions. Affinities of pairs of neighboring voxels virtually increase the resolution (b): The boundary (dotted lines) is now defined as low affinity (black squares) in the virtual grid between the original voxels.

two voxels are part of the same connected component. Affinity graphs virtually increase the resolution because the boundary is now between voxels instead of on the voxel grid (figure 15). This is particularly useful for anisotropic TEM with low resolution along the axial dimension. The concept of affinity graphs can be trivially generalized to arbitrary neighborhoods with *long-range* or *multi-range* connectivity.

Most successful neuron reconstruction approaches follow this scheme (Jain et al. 2007; Turaga, Kevin L Briggman, et al. 2009; Dan Cireșan et al. 2012; Ronneberger, Fischer, and Brox 2015; Fakhry, Peng, and Ji 2016; Beier et al. 2017; Parag et al. 2017). *Flood-filling networks* (Januszewski et al. 2016) — to the best of my knowledge, the only noteworthy exception — skip boundary prediction and directly generate fragments from raw data. Their computational cost, however, is prohibitive for large-scale use in most labs outside tech-giants like Google. Luther and Seung (2019) learn local feature-descriptors instead of boundaries but the general reconstruction workflow remains the same. As indicated above, the final step of any automatic reconstruction workflow is manual curation of the segmentation result. This can be a tedious procedure: The most severe segmentation errors — under-segmentation at the fragment level — can only be corrected for by manipulation of individual voxels. As a result, reconstruction workflows are usually tuned to encourage over-segmentation and suppress under-segmentation.

In order to give the reader a basic understanding of the algorithms and models that I will use, I will begin with a brief introduction to machine learning and relevant artificial neural network architectures, followed by a survey of related work. I will start by explaining the broader concept of supervised machine learning and how it can be used to automate the dense reconstruction of neurons from micgrographs. Then, I will introduce a novel machine learning model for the dense isotropic reconstruction of neurons from anisotropic electron micrographs.

3.2 SUPERVISED MACHINE LEARNING

In a classical computer program, the programmer instructs the computer how to translate the input into output. It is the programmer's responsibility that all possible configurations of inputs produce an appropriate and correct output value. With a sufficiently complex task, the number of possible input configurations becomes intractable and overwhelming and it can be impossible to write such a program. In supervised machine learning, in contrast, the programmer specifies a parameterized model (function)

$$\begin{aligned} f_\theta : \mathcal{X} &\mapsto \mathcal{Y} \\ x &\mapsto f_\theta(x) \end{aligned} \tag{21}$$

that maps the input space $\mathcal{X}$ into the output space $\mathcal{Y}$, where θ are the parameters of the model. f_θ can be an arbitrarily complex function with millions of parameters. The parameters θ of the model are then inferred from a finite sample of known input-output-pairs $\{(x, y) \in \mathcal{X} \times \mathcal{Y}\}$ — the *ground truth* or *training* data — by minimizing a scalar *loss* L that penalizes deviation of predicted outputs $\hat{y} = f_\theta(x)$ from the ground truth:

$$\hat{\theta} = \arg\min_\theta L(y, \hat{y}) \tag{22}$$

In other words, the exact parameterization of the model is not determined by the expertise and knowledge of the programmer; instead, the parameterization of the model is *learned* from the provided ground truth data. In that sense, machine learning is an *optimization* problem. A common optimization framework for the optimization of differentiable objective functions over the real domain without closed-form solution are *gradient descent* (Barber 2012, appendix A.6) or variants thereof.

Computer vision is a field in computer science that is occupied with the acquisition, processing, analysis, and *understanding* of two- or multi-dimensional digital images. Machine learning has been crucial for image understanding, and — as neuron segmentation ultimately is an image understanding task — is thus the logical choice of tool for neuron reconstruction. The inputs X of a computer vision machine learning task are n-dimensional arrays, sometimes referred to as *tensors* (Abadi et al. 2016). The output depends on the prediction task, *e.g.* a single vector of class probabilities for *image classification*, an n-dimensional tensor for voxel-wise boundary classification, or an $n + 1$-dimensional tensor for the prediction of affinity graphs or local feature vectors.

3.2.1 *Artificial Neural Networks*

In the past decade, artifical neural networks have outperformed other models in many computer vision and other machine learning tasks, *e.g.* Krizhevsky, Sutskever, and Hinton (2012) and Ronneberger, Fischer, and Brox (2015). In the following I will introduce basic concepts that are relevant for understanding the archicteture of the artificial neural network I will use for neuron reconstruction.

The basic computational *unit* of a neural network — in analogy to nervous systems also called a neuron — is a weighted sum of inputs $x = (x_1, \dots, x_n)$ and bias b followed by non-linear *activation function* f:

$$\begin{aligned} s(x) =& f(\sum_{i=1}^{n} w_i x_i + b) \\ =& f(\mathbf{w}^T \mathbf{x} + b) \end{aligned} \tag{23}$$

with parameters (w, b). Typically, multiple units are organized in *layers*. Units within the same layer are not connected and have the same activation function. The output of a layer can then be summarized with matrix notation:

$$\mathbf{s}(\mathbf{x}) = f(W\mathbf{x} + \mathbf{b}) \tag{24}$$

with parameters $(W, \mathbf{b})$. The non-linear function f is applied to each element of $W\mathbf{x} + \mathbf{b}$. Commonly used activation functions are the *sigmoid* function

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{25}$$

the *tangens hyperbolicus*

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{26}$$

or the *rectified linear unit* (ReLU) function

$$f(x) = \max\{0, x\}. \tag{27}$$

In some cases it is useful to omit the non-linearity and use identity (or linear) activation

$$f(x) = x. \tag{28}$$

In *feed-forward neural networks*, information flows in only one direction (forward): Data is passed from the *input* layer to an arbitrary number of successive *hidden* layers, and finally to the *output* layer. In other words, the computation graph does not contain any loops and is a directed acyclic graph. Without any non-linear activation, a neural network is just a composition of linear functions, *i.e.* a linear function, and cannot describe any non-linear input-output relations.

Typically, backpropagation (Widrow and Lehr 1990; LeCun et al. 1989) is used for learning the parameters, and thus the activation function of each unit must have a non-vanishing gradient (at least for some of the domain). A network is considered *deep* if it has at least two non-linear layers. For networks with a non-linear output layer, one non-linear hidden layer is sufficient to meet this criterion. In practice, however, feed-forward neural networks with many hidden layers and millions of parameters are used. The design or layout of the network, *i.e.* the number of layers, the sequence of layers, connectivity between layers, and other constraints, are referred to as the *architecture* of a neural network.

In the following, I will only consider feed-forward neural networks with tensors as inputs and outputs and use the terms (artificial) neural networks and feed-forward neural networks interchangeably. I will describe the classes of layers that I used for the architecture of the quasi-isotropic network (figure 21). A schematic (figure 16) visualizes a neural network with various layer types.

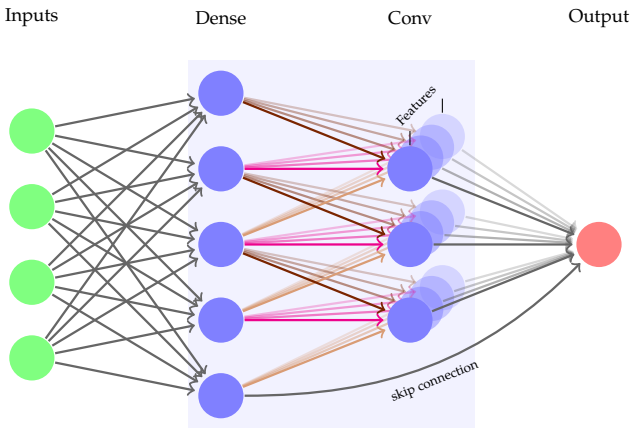


Figure 16: Neural network with two hidden layers as indicated by the shaded area. Units are indicated by circles. The explicit visualization of the activation function is omitted for ease of notation. Each unit of the dense layer is connected to all inputs. The convolutional layer (*Conv*) connects only sparsely and has shared weights as indicated by the colors of the arrows. Multiple features are indicated by transparent units and arrows. A single skip connection is indicated by the arrow from a unit in the dense layer to the output. Note that usually complete layers/features are passed as skip connections instead of just a single unit from that layer. The different layers and connection types are described in detail in sections 3.2.1.1 to 3.2.1.5

3.2.1.1 *Fully Connected*

All units of a *fully connected* or *dense* layer take the outputs of all units in the previous layer as input. The parameters $(W, \mathbf{b})$ are unconstrained and the learned weights and biases can be different for all units in the layer. Fully connected layers have a large number of parameters and are typically used in the final layers of image classification networks.

3.2.1.2 *Pooling*

Pooling layers are used to sub-sample the input tensor and thus reduce the number of parameters: A single output is selected from a small field-of-view within the input tensor with respect to a specific criterion, *e.g.* maximum or average activation (Serre et al. 2007; Scherer, Müller, and Behnke 2010). Typically, the fields-of-view do not overlap in input space. This reduces over-fitting, extracts representative features from the input tensor, and increases the receptive field, in particular for purely convolutional (section 3.2.1.3) networks. The smaller number of paramaters also reduces the overall size of the architecture and allows for faster training or additional layers.

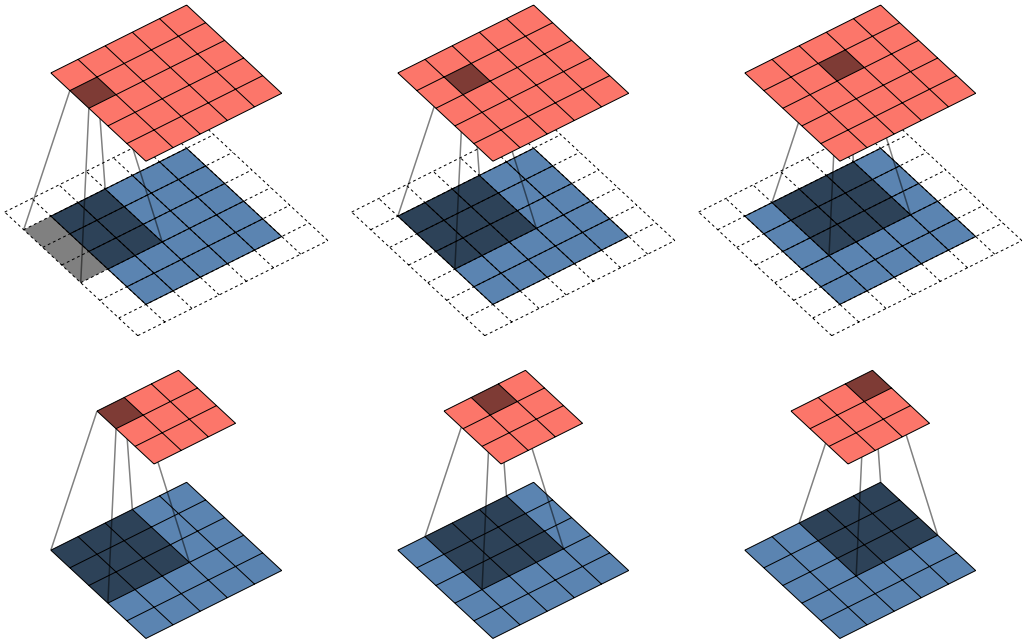


Figure 17: A two-dimensional input (blue) is convolved with a 3×3 kernel: The kernel slides over the input with unit stride and computes output (red) pixels as the weighted sum of its field-of-view in the input. The weights are speicifed by the kernel and are independent of the spatial location of the kernel. The input can be padded (top row, dashed) to produce an output of the same size. No padding (bottom row) ensures that every output pixel is generated from valid data only but reduces tensor size.

3.2.1.3 *Convolutional*

In a convolutional layer, the weights and bias are shared between units. This constrains the weight matrix, *e.g.* for one-dimensional tensors with n non-zero weights $w_1, w_2, \dots, w_n$,

$$W = \begin{pmatrix} w_1 & w_2 & \cdots & w_n & 0 & 0 & 0 & \cdots \\ 0 & w_1 & w_2 & \cdots & w_n & 0 & 0 & \cdots \\ 0 & 0 & w_1 & w_2 & \cdots & w_n & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}. \tag{29}$$

Optionally, the bias $\mathbf{b}$ can be omitted or shared among units:

$$b_i = b_j \; \forall \; i, j. \tag{30}$$

The constrained matrix W describes a discrete cross-correlation of a function g with kernel k:

$$(g \star k)[n] = \sum_{m=-\infty}^{\infty} g[m] \times k[m+n] \tag{31}$$

$$= \sum_{m=-\infty}^{\infty} g[m-n] \times k[m] \tag{32}$$

Cross-correlation is equivalent to convolution with the reflection $\tilde{g}[m] = g[-m]$ of g:

$$(g \star k)[n] = \sum_{m=-\infty}^{+\infty} g[m] \times k[m+n] \tag{33}$$

$$= \sum_{m=+\infty}^{-\infty} g[-m] \times k[-m+n] \tag{34}$$

$$= \sum_{m=+\infty}^{-\infty} \tilde{g}[m] \times k[n-m] \tag{35}$$

$$(g \star k)[n] = (\tilde{g} * k)[n] \tag{36}$$

By convention, cross-correlation is referred to as convolution in the context of convolutional neural networks and I will follow this convention throughout this dissertation.[2]

Typically, kernels have a limited field-of-view with $2j + 1$ non-zero entries. Only non-zero entries of the kernel are considered in this summation.

$$(g \star k)[n] = \sum_{m=-j}^{j} g[m-n] \times k[m] \tag{37}$$

Conceptually, the kernel is shifted over the entire input at integral strides. The output at each location is the weighted sum of the local field-of-view (figure 17). Appropriate padding preserves the size of the input tensor. No (or *valid*) padding reduces the tensor size but only valid data is processes.

[2]The reason for the choice of nomenclature is unclear. The use of convolutions in digital signal and image processing may be an explanation.

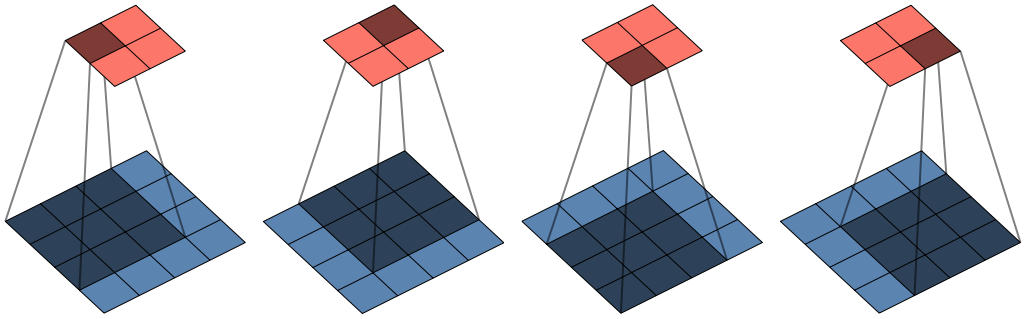


Figure 18: 2D convolution of a 4×4 sized input with a kernel of size 3×3 into a 2×2 output. All steps are visualized.

Multiple kernels can be a combined in a single convolutional layer: The outputs of the individual kernels are concatenated as channels of a single tensor, usually called a *feature* map. Early convolutional layers in a neural network are usually low-level image features that sometimes resemble classical hand-crafted digital image processing features like edge detectors (Zeiler and Fergus 2014). Convolutional layers at a later stage in the network produce higher-level features.

Possible extensions of convolutional layers include non-unit strides for learned down-sampling similar to pooling (section 3.2.1.2) and dilated convolutions (Yu and Koltun 2016) for an increased receptive field without increasing the number of parameters. The kernel considers only every l^{th} entry of the input:

$$(g \star_l k)[n] = \sum_{m=-\infty}^{\infty} g[n + l \times m] \times k[m] \tag{38}$$

3.2.1.4 *Transposed Convolutions*

In a fully convolutional setting with downsampling layers (pooling, non-unit stride convolutions) it can be necessary to upsample a layer. Instead of using engineered interpolation methods (nearest neighbor, linear, cubic), *transposed* convolutions learn the correct way to up-sample the data. The field-of-view in the output tensor corresponds to a single voxel in the input tensor. The kernel is scaled by the value of the input voxel and added to the field-of-view in the output tensor. Unrolling the 2D input and output tensors of figure 18 into 16-dimensional and 4-dimensional vectors, respectively, the convolution operation can be formulated as mulitplication with a sparse matrix

$$K = \begin{pmatrix} k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} & 0 \\ 0 & 0 & 0 & 0 & 0 & k_{1,1} & k_{1,2} & k_{1,3} & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & k_{3,1} & k_{3,2} & k_{3,3} \end{pmatrix}$$

Here, the kernel is centered at $k_{2,2}$ for ease of notation. During training, errors are propagated by mulitplication of the loss with the transpose K^T: The kernel k defines both K and K^T and thus a transformation from 2×2 input to a 4×4 output with compatible connectivity. Consequently, a transposed convolution

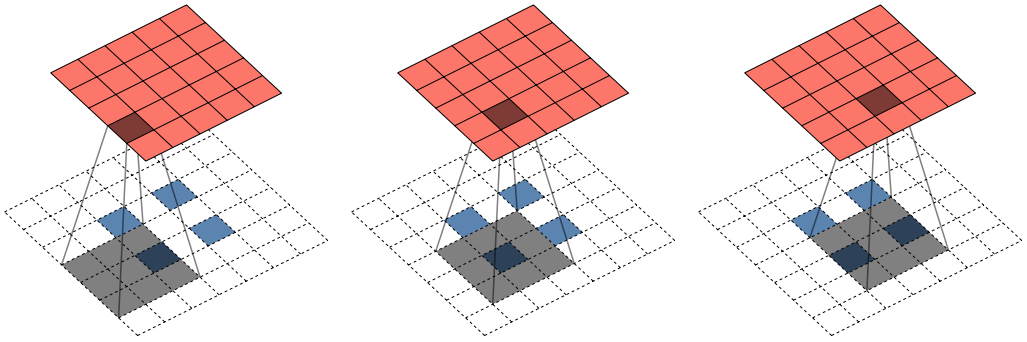


Figure 19: The 2D convolution of a 2×2 sized input with a kernel of size 3×3 into a 5×5 output with unit stride, 2×2 border padding, and 1×1 padding between input voxels is an equivalent representation of the transpose of convolving an input 5×5 with a 3×3 kernel and 2×2 stride.

swaps forward and backward passes of a convolution matrix (Dumoulin and Visin 2016). A transposed convolution has an equivalent representation by a convolution with appropriate kernel size, padding between input pixels, border padding, and stride (figure 19).

Other common names are fractionally strided convolutions, inverse, up, or backward convolutions, or deconvolutions (Shi et al. 2016). The latter is misleading as transposed convolutions are not the inverse operation of convolutions. Throughout this dissertation, I will therefore use the term transposed convolutions.

3.2.1.5 *Skip Connections*

Features from early layers can provide meaningful information for layers much deeper in the network but the information is heavily processed by non-linear layers in between. *Skip* or *copy-crop* (Funke, Tschopp, et al. 2018) connections bypass intermediate layers and feed the output of layer k directly to layer $j \geq k + 2$. The outputs of layer k are concatenated with other inputs for layer j. Intermediate layers may change the size of tensors and consistency is ensured by cropping. In networks with down-sampling (pooling, non-unit stride convolutions) and up-sampling (transposed convolutions), naive skip connections are only possible between layers that have the same voxel size.

3.3 RELATED WORK

Automatic neuron reconstruction approaches have been following a multi-step workflow as outlined in section 3.1. Methods typically differ in the algorithms that are used for the prediction of boundaries from raw data, the generation of fragments from boundary predictions, and the agglomeration of fragments into segments that ideally coincide with the neurons present in the raw data. As a general rule of thumb, higher quality boundary predictions and fragments simplify both the agglomeration task and subsequent proof-reading.

Supervised machine learning was first used for boundary classification in EM by Jain et al. (2007): A three-dimensionl (purely) convolutional neural network with 34041 free parameters outperformed markov random fields and conditional random fields for pixel classification. Björn Andres et al. (2008) extended that work in two ways: They used computationally cheaper random forest classifiers (Breiman 2001) instead of a neural network without loss of performance and they introduced an agglomeration scheme as described in section 3.1 and figure 14: Fragments from seeded watersheds on the boundary map are merged based on weights learned from another random forest classifier.

Turaga, Murray, et al. (2010) were the first to use neural networks to learn affinity graphs for neuron reconstruction from ground truth annotations. Together with their minimax loss (Turaga, Kevin L Briggman, et al. 2009), they can directly minimize the rand index (Rand 1971) during training. They argue that learning better affinities will allow for more simple agglomeration or graph partitioning.

Dan Cireșan et al. (2012) adapted image classification neural networks (D. Cireșan, Meier, and Schmidhuber 2012) with max-pooling layers (Serre et al. 2007; Scherer, Müller, and Behnke 2010) for pixel-wise membrane probability prediction in two-dimensional images of electron micrographs. The outputs of multiple networks are averaged to reduce variance. Contrary to convolutional networks, the network predicts the probability for a single voxel only: The receptive field for each voxel prediction is increased at the cost of longer inference times during prediction.

Targeting anisotropic TEM section series, Funke, Bjoern Andres, et al. (2012) generated multiple two-dimensional segmentation hypotheses within sections from different parameterizations of the graph-cuts algorithm (Boykov and Kolmogorov 2004). A graphical model was formulated with consistency constraints and costs for linking hypotheses across sections. Costs were predicted with a random forest classifier. The optimal linkage and segmentation was found with integer linear programming. Similarly, Kaynig et al. (2015) fused two-dimensional segmentation hypotheses Vazquez-Reina et al. (2011) from random forest boundary predictions. The similarity to the cell tracking problem has inspired similar models in that domain (Schiegg et al. 2014).

Seyedhosseini, Sajjadi, and Tasdizen (2013) increased context by cascading multiple classifiers. T. Liu et al. (2014) used boundary predictions from Dan Cireșan et al. (2012) and Seyedhosseini, Sajjadi, and Tasdizen (2013) to generate two-dimensional fragments from morphological watersheds (Beare and Lehmann 2006) and then learned the weights of hierarchical merge trees with random forest classifieres for agglomeration in 3D.

Nunez-Iglesias, Kennedy, Parag, et al. (2013), Nunez-Iglesias, Kennedy, Stephen M Plaza, et al. (2014), and C. Jones et al. (2015) extended hierarchical agglomeration with an active-learning scheme: The weights were learned from expert annotations during an interactive proof-reading process. The segmentation was updated and presented to the user on-the-fly.

Similar to Funke, Bjoern Andres, et al. (2012), Kaynig et al. (2015) fused two-dimensional segmentation hypotheses Vazquez-Reina et al. (2011) from random forest boundary predictions. Uzunbas, C. Chen, and Metaxas (2016) extended

that idea by guiding proof-readers to locations of uncertainty of the agglomeration model.

Ronneberger, Fischer, and Brox (2015) introduced a new neural network model that utilizes multiple scales and drastically extends the field of view for pixel-wise boundary classification, the *U-net*: A cascade of convolutional and down-sampling layers increases the field-of-view and captures context. Alternated convolutions and up-sampling then enable precise localization. Upsampled layers are connected to their respective counterparts in the down-sampling branch via skip connections. The use of extensive data augmentations virtually increases the limited amount of available training data. Çiçek et al. (2016) extended the U-net to three spatial dimensions.

Funke, Tschopp, et al. (2017) and Funke, Tschopp, et al. (2018) trained a 3D U-net for affinity graphs (Turaga, Murray, et al. 2010) with a more efficient revision of Turaga, Kevin L Briggman, et al. (2009). They argued that, due to signficantly lower z-resolution in anisotropic TEM, voxel-wise boundary classification cannot separate all objects; affinity graphs virtually increase resolution to solve this problem. Two-dimensional fragments are extracted for agglomeration. Even with simple percentile-based agglomeration, they outperform existing models. The U-net has been adapted successfully for other 3D-EM prediction tasks, *e.g.* synaptic cleft reconstruction (Heinrich, Funke, et al. 2018). The latter extended Funke, Tschopp, et al. (2017) for application in anisotropic TEM micrographs: Modified kernel sizes ensure near-isotropic field-of-views for most layers without a change to the overall design. They showed that a near-isotropic field of view boosts performance (figure 20).

Beier et al. (2017) use multi-cut agglomeration after long-range affinity prediction with cascaded random forests or neural networks to outperform previous work. Following the argument of Funke, Tschopp, et al. (2017) and Funke, Tschopp, et al. (2018), fragments are two-dimensional.

Most recently, Luther and Seung (2019) adapted deep metric learning for semantic image segmentation (De Brabandere, Neven, and Van Gool 2017; Fathi et al. 2017) to neuron reconstruction. Similarly, Sheridan et al. (in preparation) learn local shape descriptor features.

Building on this body of previous work, in particular (Çiçek et al. 2016; Funke, Tschopp, et al. 2017; Funke, Tschopp, et al. 2018; Heinrich, Bogovic, and Saalfeld 2017; Heinrich, Funke, et al. 2018; Beier et al. 2017), I will present a novel neural network architecture for the prediction of multi-range affinity graphs at quasi-isotropic resolution from anisotropic TEM.

3.4 QUASI-ISOTROPIC NETWORK ARCHITECTURE

Recent serial section TEM (Zheng et al. 2018) has a nominal spatial resolution of $4 \times 4 \times 40\,\mathrm{nm}^3$. As a result of this significant anisotropy, boundary predictions cannot capture rapid change of cross-sections of individual neurons along the axial dimension and under-segmentation errors are a common observation in 3D fragments. Under-segmentation at the fragment level cannot be recovered during agglomeration and state-of-the-art neuron reconstruction workflows (Beier et al.

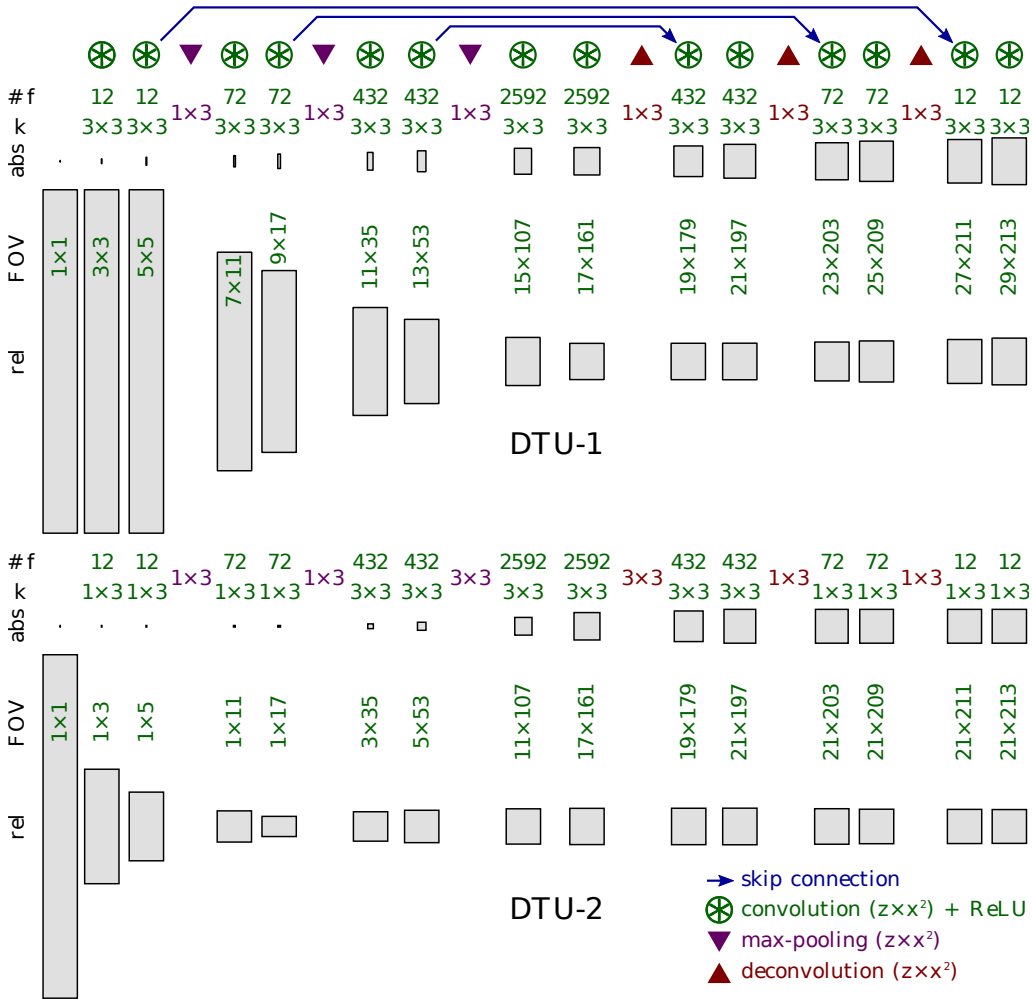


Figure 20: 3D U-net architectures for synaptic cleft predictions (taken from Heinrich, Funke, et al. (2018)). Isotropy of x and y allows to collapse both dimensions into the second dimension in the sketch without loss of expressiveness. Each layer is annotated with the field-of-view (FOV) in units of input voxels. Absolute and relative FOVs under consideration of the voxel sizes are visualized as rectangles. A square shape means perfect isotropy. While the FOV remains anisotropic for all of the downsampling pathway of DTU1, DTU2 approaches near-isotropic FOV early in the downsampling pathway.

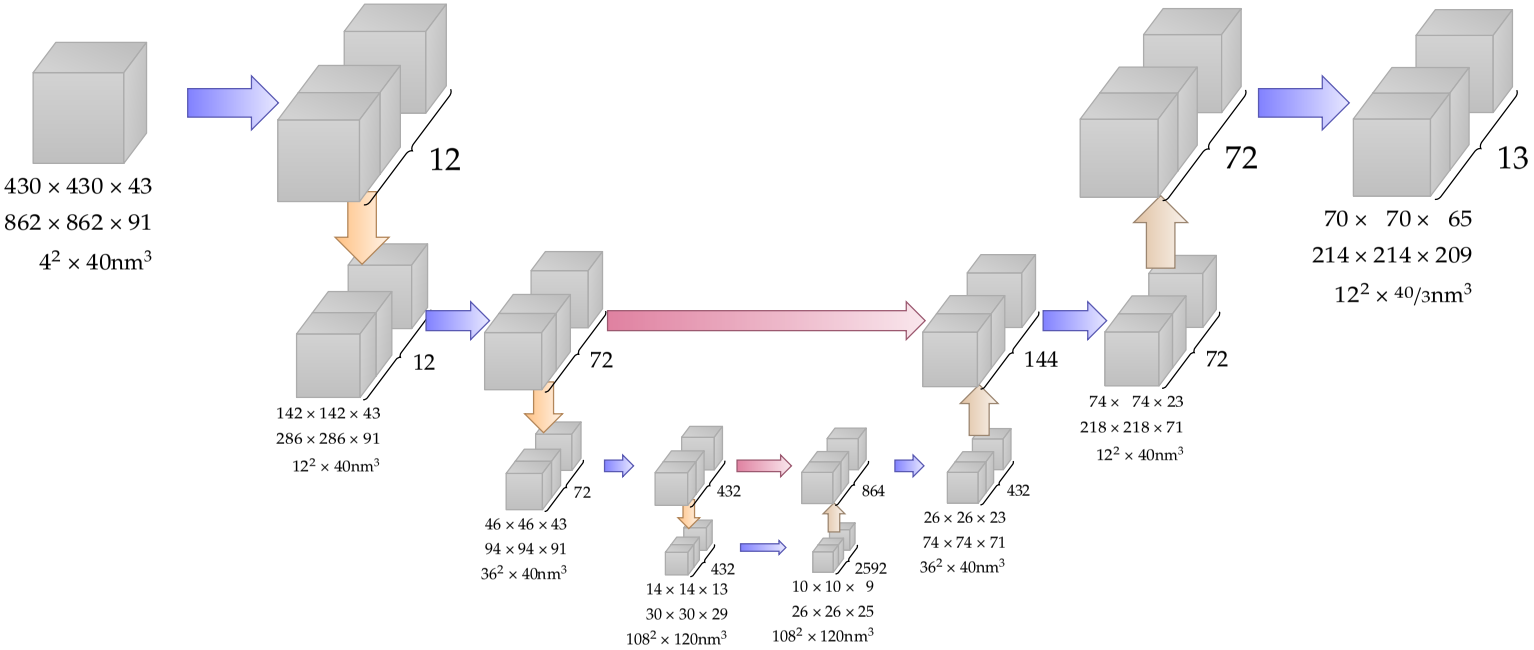


Figure 21: Quasi-isotropic network architecture: Arrow color encodes neural network connectivity types between tensors (gray blocks): blue stands for two consecutive convolutional layers, yellow for down-scaling with pooling, red for skip connections, and brown for up-sampling with transposed convolutions. Tensor sizes for training (first row) and inference (second row) as well as voxel sizes (third row) are specified below the tensors, nubmer of features with curly braces. The number of output features is the sum of predicted affinities N and a single glial cell channel: $N + 1$. I trained networks with 12 affinity channels in my experiments and thus the number of output features is 13. Note that the activation function after the final convolution is identity, *i.e.* it is not non-linear. Tensors are not visualized to scale. Modified from Funke, Tschopp, et al. (2018) with many thanks to Jan Funke for sharing the code.

2017; Funke, Tschopp, et al. 2018) thus extract 2D fragments within individual xy-planes and agglomerate these fragments in three dimensions. The larger number of 2D fragments compared to 3D fragments can be a limiting factor for complex agglomeration schemes and proof-reading.

Naturally, larger 3D fragments can be extracted from boundary predictions at high isotropic resolutions. I propose a novel neural network architecture to create boundary predictions at high, quasi-isotropic resolution from anisotropic TEM with low resolution along the axial dimension.

TEM oversamples the data along the higher-resolution x and y dimensions for the purpose of neuron reconstruction. It is thus possible to achieve *quasi-isotropic* resolution by down-sampling x- and y-dimensions by a factor of 3 and up-sampling the z-dimension by a factor of 3 for a spatial resolution of $12 \times 12 \times {}^{40}/_{3}\,\mathrm{nm}^3$. Down-sampling in x and y reduces the amount of data but should not impede neuron reconstruction as the down-sampled resolution of 12nm is still significantly finer than relevant processes.

I created a novel U-net like architecture based on Heinrich, Funke, et al. (2018) to predict quasi-isotropic affinity graphs at a resolution of $12 \times 12 \times {}^{40}/_{3}\,\mathrm{nm}^3$. Figure 21 shows a sketch of my quasi-isotropic architecture: Two convolutional layers (section 3.2.1.3) with kernel size $3 \times 3 \times 1$, 12 features and ReLU (equation (27)) activation function are followed by a pooling layer (section 3.2.1.2) with kernel size $3 \times 3 \times 1$ for a voxel size of $12 \times 12 \times 40\,\mathrm{nm}^3$. This is followed by a U-net with three levels of resolution with symmetric down-sampling and up-sampling with factors $3 \times 3 \times 1$ (for a voxel size of $36 \times 36 \times 40\,\mathrm{nm}^3$) and $3 \times 3 \times 3$ ($108 \times 108 \times 120\,\mathrm{nm}^3$). Two consecutive convolutional layers with ReLU activation preceed each down-sampling and succeed each up-sampling layer. The number of features increases by a factor of 6 with each level of the U-net. Skip connections (section 3.2.1.5) connect down-sampling path and up-sampling path for all but the bottom layer of the U-net. At the end of the U-net, the voxel size is $12 \times 12 \times 40\,\mathrm{nm}^3$. An additional transposed convolution (section 3.2.1.4) brings the voxel size to quasi-isotropic $12 \times 12 \times {}^{40}/_{3}\,\mathrm{nm}^3$, followed by a convolutional layer with kernel size $3 \times 3 \times 3$, 12 features, and ReLU activation, and a convolutional layer with kernel size $1 \times 1 \times 1$ and identity (equation (28)) activation to predict 12 affinity channels and glial cell probability (section 3.4.1.1).

The key change in network architecture is to up-sample the z-dimension by a factor of 3 instead of the x and y dimensions in the final up-sampling layer of the network. As a result, skip connections cannot be used for that layer, as the voxel size is inconsistent with the corresponding layer in the downsampling pathway. Figure 22 visualizes the absolute and relative fields-of-view at each layer of the network in the same way as figure 20.

I implemented the network architecture in the *Python*[3] programming language using the *Gunpowder*[4] framework for machine-learning on multi-dimensional images with tensorflow[5] (Abadi et al. 2016) as backend. I adapted the U-net im-

[3] https://www.python.org
[4] https://github.com/funkey/gunpowder
[5] https://www.tensorflow.org/

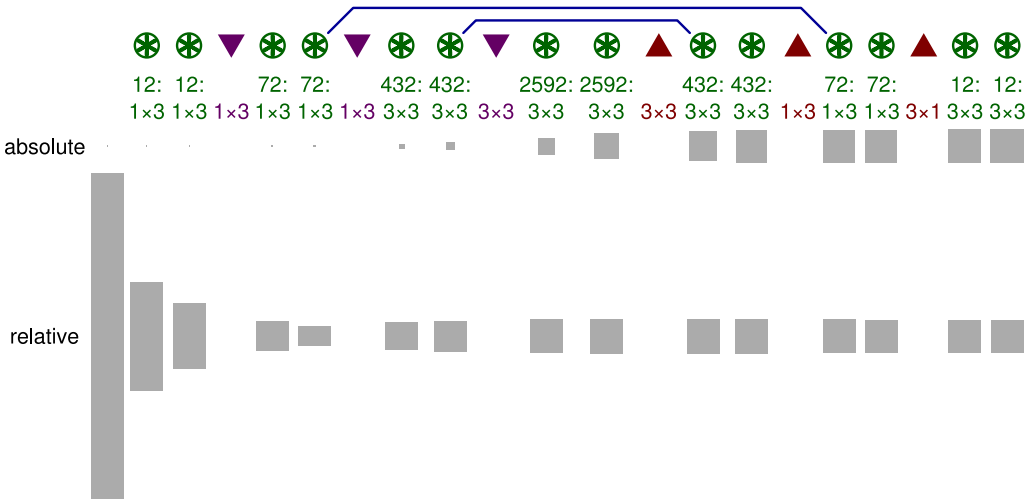


Figure 22: Quasi-isotropic network architecture is an extension of DTU2 in figure 20: The last up-sampling is along the z-dimension instead of the x- and y-dimensions. Note that the outermost skip connection had to be dropped because voxel sizes are inconsistent at the corresponding levels. Legend: see figure 20.

plementation from Larissa Heinrich's *CNNectome* repository[6]. I implemented additional Gunpowder nodes that were required for my new network architecture in the *fuse*[7] Python package. The network architecture and management of reproducible experiments is available through the *eqip*[8] Python package (section 4.2).

3.4.1 *Ground Truth*

In general, powerful modern convolutional neural networks have millions of parameters and good performance can only be achieved with a sufficient amount of densely annotated ground truth training data. Ground truth generation is a time-consuming process and, thus, the amount of publicly available ground truth data is limited. To address this problem, the CREMI challenge[9] provided an unprecedented wealth of densely annotated ground truth: three pairs (training and test data) — identified as samples A, B, and C; "+" indicates test data — of $(5\mu m)^3$ or $(1250 \times 1250 \times 125 px^3)$ cubes from different regions of the adult fly brain with nominal anisotropic voxel size $4 \times 4 \times 40\,nm^3$, totaling to 1171875000 voxels (or about 1.2 gigavoxels) of annotated data. The training ground truth data is publicly available through the challenge website. Scott Lauritzen, Arlo Sheridan and Constantin Pape proof-read predictions from Constantin Pape on three similarly-sized blocks from the same adult fly brain identified as samples 0, 1, and 2. They also proof-read samples A, B, and C. I use 75% of proof-read samples 0, 1, 2 and realigned samples A, B, C for training. The remaining 25%

[6] https://github.com/saalfeldlab/CNNectome
[7] https://github.com/saalfeldlab/fuse
[8] https://github.com/saalfeldlab/eqip
[9] https://cremi.org

are used for evaluation. The ground truth is annotated at the same anisotropic spatial resolution as the raw data: $4 \times 4 \times 40\,\mathrm{nm}^3$.

3.4.1.1 *Glial Cells*

The central nervous system is intermingled by glial cells — non-neuronal cells that can be similar in appearence to neurons in 3D-EM. Instance segmentation of glial cells is susceptible to under-segmentation errors because individual glial cells are hard to identify. The effect on neuron reconstruction can be catastrophic in combination with confusion of glial cells with neurons: Prohibitively large numbers of neurons are falsely merged.

I implemented the following strategy to minimize glial cell related under-segmention: The separation of glial cells into individual instances with distinct labels is not relevant for neuron reconstruction. An additional dedicated label for glial cells in the ground truth data encourages zero affinity between neurons and glial cells. Optionally, affinities within glial cells can be ignored for calculating the loss during training. The additional semantic segmentation task for the detection of glial cells employs the benefits of multi-task learning (Caruana 1997): Individual tasks can inform each other about the inherent structure of the data or a task through shared representations. Furthermore, the predicted glial cell mask can be used to restrict super-voxels and avoid undersegmentation caused by confusion of glial cells with neurons at prediction time. Glial cell annotations were generated in a combined effort by Arlo Sheridan, Constantin Pape, Scott Lauritzen, and Jan Funke.

3.4.1.2 *Label Interpolation*

While the CREMI challenge provides a wealth of densely annotated ground truth data for neuron reconstruction from TEM, ground truth data at a quasi-isotropic resolution is not available. The generation of dense ground truth data is laborious and manual interpolation of ground truth data is challenging at best. Therefore, I developed a simple yet effective distance-transform based label interpolation method in order to generate quasi-isotropic ground truth data from anisotropic annotations. Given a regular grid $\mathcal{G}$ and a sampled function, *i.e.* an image,

$$\begin{aligned} f : \mathcal{G} &\mapsto \mathbb{R} \\ p \in \mathcal{G} &\mapsto f(p) \end{aligned} \tag{39}$$

and a distance d, Felzenszwalb and Huttenlocher (2012) define the distance transform for $p \in \mathcal{G}$

$$\mathcal{D}_f(p) = \min_{q \in \mathcal{G}} d(p,q) + f(q). \tag{40}$$

Then, the binary distance transform of a set of points $P \subseteq \mathcal{G}$ is a special case

$$\mathcal{D}_P(p) = \min_{q \in \mathcal{G}} d(p,q) + \mathbb{1}_P(q), \tag{41}$$

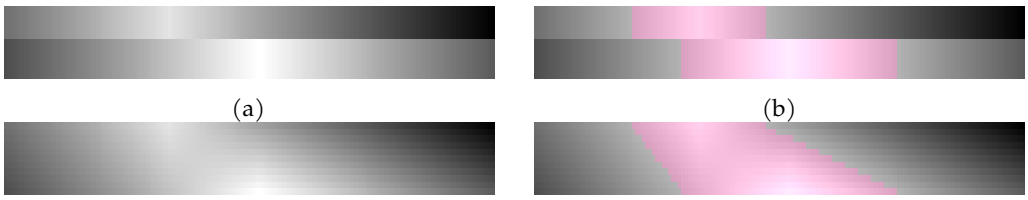


Figure 23: Distance transform interpolation between two thick, one-dimensional sections: Signed distance transforms in thick sections (a) are thresholded at zero to create associated objects (b). After interpolation of the distance transforms (c), the transition between objects in adjacent sections appears smoother (d). The contrast range for the distance transform is $[-79, 33]$.

where the indicator

$$\mathbb{1}_P(q) = \begin{cases} 0 & \text{if } q \in P, \\ \infty & \text{otherwise.} \end{cases} \tag{42}$$

I calculate the binary distance transform for all labels $L_n = \{l : f^n(p \in S_n) = l\}$ of the n^{th} section S_n of an annotaed series of N sections:

$$\{D^n_{P_l} : \forall l \in L_n\} \tag{43}$$

Then, for all ordered pairs of successive sections $n, n+1 : 1 \leq n < N$, k intermediate sections are generated at equi-distant spacing between sections n and $n+1$ by interpolating the distance-transforms of each section with appropriate weights for each label, and using a label for a pixel if the interpolated signed distance transform is smaller or equal to zero. The number of intermediate sections k is chosen such that the resulting voxel resolution is (nearly) isotropic. figure 23 provides an example for one-dimensional sections. For a resolution of $4 \times 4 \times 40\,\text{nm}^3$ a value of

$$k = 9 \tag{44}$$

would result in an isotropic resolution that would highly oversample the data for neuron reconstruction. Therefore, the data is downsampled by a factor of 3 along each of x and y and a choice of

$$k = 2 \tag{45}$$

achieves a quasi-isotropic voxel resolution $12 \times 12 \times \frac{40}{3}\,\text{nm}^3$ and reduces the amount of total voxels by a factor of $\frac{3\times3}{3} = 3$ with respect to the original data. In particular, the costly computation and interpolation of distance transforms for all labels in pairs of sections is not executed as many times.

3.4.1.3 *Limitations of Distance Transform Interpolation*

Interpolation of the distance transform is an approximation of interpolation of objects that does not explicitly model objects. As a result, the size of interpolated objects in interpolated sections is underestimated when there is little overlap across

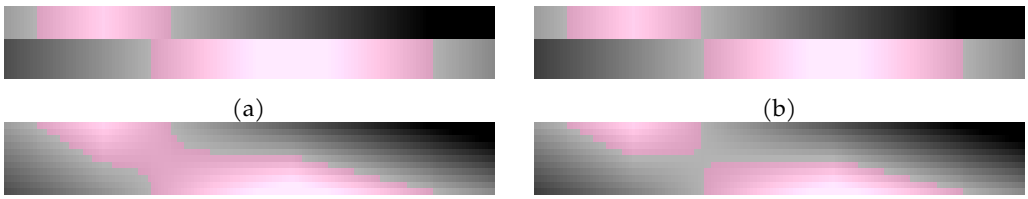


Figure 24: Failure modes of label data interpolation by means of distance transform interpolation: Objects that have only very small overlap across sections (a) are thinned out by distance transform interpolation (c). In the extreme case of no overlap (b), the interpolation will create two separate connected components that shrink and finally disappear (d).

Table 3: Quasi-isotropic network settings: # MSE and # Malis denote the number of training iterations with mean squared error and malis losses, respectively. A check-mark (cross-mark) in the "Intra-Glial Loss" column indicates that affinities within glial cells are considered (ignored) in the loss.

Network	# MSE	# Malis	Intra-Glial Loss
qi-mse	450,000	0	✓
qi-mse-ng	450,000	0	✗
qi-mls	0	450,000	✓
qi-mls-ng	0	450,000	✗
qi-mls-pre	20,000	430,000	✓
qi-mls-pre-ng	20,000	430,000	✗

sections (figure 24 left column) or — in the extreme case of no overlap (figure 24 right column) — the interpolated objects are disconnected into two separate connected components, favoring over-segmentation. While over-segmentation is the preferred failure mode over under-segmentation, a more appropriate interpolation mode would decrease label noise and improve reconstruction accuracy. The most simple extension of a distance transform interpolation would be to ensure that the objects are concentric by some measure, *e.g.* center of mass, and then interpolate the distance transforms of the centered objects and translate the interpolated objects by the interpolation of the original centers of mass. This can be very effective for rigid, compact objects that are only translated across sections (figure 25) but fails to capture rotations or any other non-translational transformations of the object across sections, and it is easy to construct an example with little or no overlap of the concentric objects, *e.g.* annuli. Instead, representations of objects as two-dimensional polygons can be easily interpolated by matching vertices of the polygons and then simply interpolating the positions of the vertices. These optimizations of the label interpolation method will be the subject of future work.

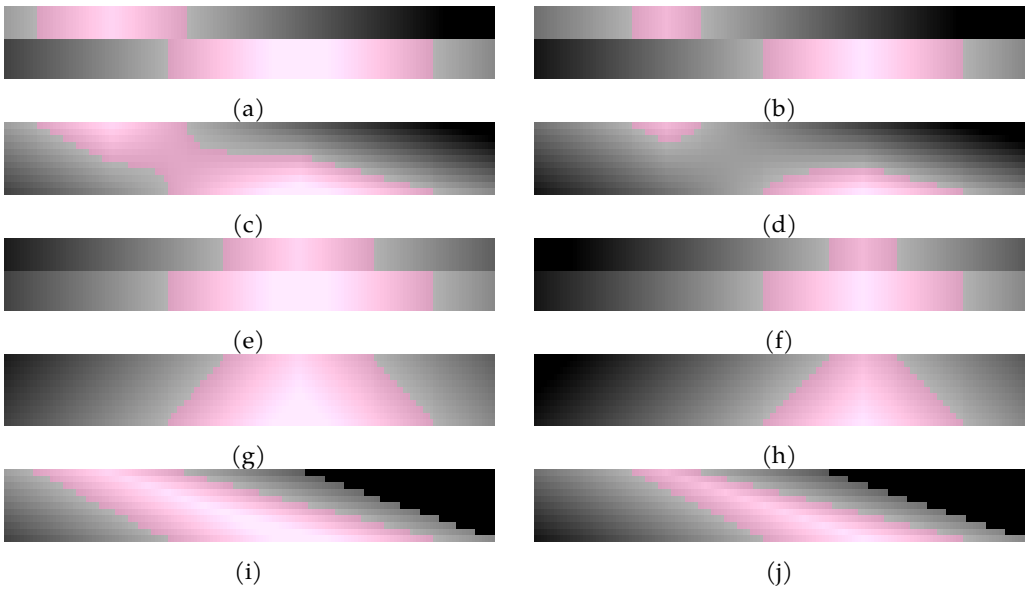


Figure 25: Aligning objects by their centers-of-mass can alleviate narrowing (c) or vanishing (d) objects in little (a) or no overlap (b) situations: First, the objects are aligned by their respective centers-of-mass (e)(f). Then, distance transforms are interpolated as before (g)(h). Finally, each section section is translated such that each of the original objects are in the crorrect place (i)(j).

3.4.2 *Experiments*

I trained a total of six networks with quasi-isotropic (qi) architecture with three different losses, and ignoring or considering loss within glial cells (section 3.4.1.1): mean squared error loss considering (qi-mse) and ignoring (qi-mse-ng) intra-glial loss, malis loss (Turaga, Kevin L Briggman, et al. 2009) considering (qi-mls) and ignoring (qi-mls-ng) intra-glial loss, and malis loss, pre-trained with mean squared error, considering (qi-mls-pre) and ignoring (qi-mls-pre-ng) intra-glial loss. The network settings are summarized in table 3. Regardless of the loss type, ground truth affinities as well as glial cell labeling are balanced. The network architecture was implemented within the tensorflow framework and each of the instances was trained with the Adam optimizer (Kingma and Ba 2014) with parameters settings as shown in table 4.

The structure of the affinity graph is defined by a matrix

$$O = \begin{pmatrix} O_{1,1} & O_{1,2} & O_{1,3} \\ O_{2,1} & O_{2,2} & O_{2,3} \\ & \vdots & \\ O_{N,1} & O_{N,2} & O_{N,3} \end{pmatrix} \in \mathbb{Z}^{N \times 3} \tag{46}$$

where each row i defines an offset such that there is an edge between voxels p and $q = p + O_{i,*}$ in the affinity graph. The offsets are defined in integral units of voxels along each dimension of the data. Similarly, the affinities at voxel location p are represented by an N-dimensional real-valued vector

$$\mathbf{a}(p) = (a_1(p), a_2(p), \dots, a_N(p))^T \in \mathbb{R}^N \tag{47}$$

Assuming that the affinity graph is undirected and symmetric

$$O_{i,k} = -O_{i+N/2,k} \;\forall\; i \leq N/2, k \in \{1,2,3\} \tag{48}$$

$$a_i(p) = a_{i+N/2}(p + O_{i,*}) \;\forall\; i \leq N/2, \tag{49}$$

it is sufficient to consider only the top half of the structuring matrix

$$\tilde{O} \in \mathbb{Z}^{N/2\times 3} \tag{50}$$

$$\tilde{O}_{ik} = O_{ik} \;\forall\; i \leq N/2, k \tag{51}$$

and affinities

$$\tilde{a}(p) \in \mathbb{R}^{N/2} \tag{52}$$

$$\tilde{a}_i(p) = a_i(p) \;\forall\; i \leq N/2. \tag{53}$$

For my experiment, I use four offsets along each dimension of the data to include long-range affinities:

$$\tilde{O} = \begin{pmatrix} -1 & 0 & 0 \\ -2 & 0 & 0 \\ -5 & 0 & 0 \\ -10 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & -2 & 0 \\ 0 & -5 & 0 \\ 0 & -10 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & -2 \\ 0 & 0 & -5 \\ 0 & 0 & -10 \end{pmatrix} \tag{54}$$

Beier et al. (2017) used multi-range affinities in combination with a mean square error loss. To the best of my knowledge, I am the first to train multi-range affinities networks with the malis loss.

Binary ground truth affinities $a^*(p) \in \{0,1\}^N$ are generated from ground truth neuron label data $l(p)$ to indicate if two voxels belong to the same object:

$$a_i^*(p) = \begin{cases} 1 & \text{if } l(p) = l(p + O_{i,*}), \\ 0 & \text{otherwise.} \end{cases} \tag{55}$$

The scalar glia channel prediction at location p is denoted as $g(p) \in \mathbb{R}$ and the ground truth g^* is provided directly by a binary glia mask. While the ground truth for both affinities and glia takes binary values, values outside the unit interval $[0,1]$ are possible — for example, the values 0.8 and 1.2 have the same contribution to the loss for a ground truth value of 1.0 under mean squared error loss — and the notation

$$a_l^u(p) = \begin{pmatrix} \max\{\min\{a_1(p), u\}, l\} \\ \max\{\min\{a_2(p), u\}, l\} \\ \vdots \\ \max\{\min\{a_N(p), u\}, l\} \end{pmatrix} \tag{56}$$

$$g_l^u(p) = \max\{\min\{g(p), b\}, a\} \tag{57}$$

Table 4: Parameters for Adam optimizer used for the training of quasi-isotropic architectures: α is the learning rate, β_1 and β_2 are the decay rates for linear and quadratic momentum of the gradient, respectively, and $\hat{\epsilon}$ is a small constant to ensure numerical stability.

Parameter	α	β_1	β_2	$\hat{\epsilon}$
Value	5×10^{-5}	0.95	0.999	10^{-8}

restricts predictions to the interval $[l, u]$. For ease of notation, the argument p is omitted if possible without ambiguity, *i.e.*

$$a \equiv a(p), \tag{58}$$
$$a_i \equiv a_i(p), \tag{59}$$
$$\tilde{a} \equiv \tilde{a}(p), \tag{60}$$
$$\tilde{a}_i \equiv \tilde{a}_i(p), \tag{61}$$
$$a^* \equiv a^*(p), \tag{62}$$
$$a_i^* \equiv a_i^*(p), \tag{63}$$
$$\bar{a} \equiv \bar{a}(p), \tag{64}$$
$$g \equiv g(p), \tag{65}$$
$$g_l^u \equiv g_l^u(p). \tag{66}$$

The provided field of view of the training data extends beyond the annotated ground truth data and affinities of edges with at least one voxel outside the annotated area are ignored for training. Optionally, affinities that are entirely within the glia mask can be ignored. Note that affinities between any neuron id and glia mask are always zero and will never be ignored. Ground truth affinities and glia are balanced per training sample.

3.4.2.1 *Augmentations*

Data augmentation has been established to virtually increase the amount of training data, to minimize overfitting, and to model data defects that may be under-represented in the training samples (Krizhevsky, Sutskever, and Hinton 2012; Dosovitskiy et al. 2014; Ronneberger, Fischer, and Brox 2015; Beier et al. 2017; Funke, Tschopp, et al. 2018; Heinrich, Funke, et al. 2018). I use the following random data augmentations in experiments (availability in Gunpowder or fuse as specified):

ELASTIC (FUSE) Deform data with a low-frequency elastic transformation defined by random jitter of sparse control points and rotate by a random angle around the z-axis.

MISALIGN (FUSE) Randomly translate a random xy-section within the xy-plane: Shifts affect the section and all subsequent sections in the series, slips only affect the randomly selected section and label data can be excluded from slips.

SIMPLE (GUNPOWDER) Randomly mirror or transpose a sample. Due to the anisotropic voxel resolution of the raw data, only the x and y-axes are considered for transposition.

INTENSITY (GUNPOWDER) Apply a random scale and shift to the intensity of the data.

DEFECT (GUNPOWDER) Model various instances of imaging noise and defects by randomly introducing missing sections, low-contrast sections, or imaging artifacts from a separate artifact provider.

Training samples are queried uniformly from the six training data sets — samples A, B, C, 0, 1, and 2 — at random locations and are randomly augmented on the fly with the augmentation nodes in the Gunpowder framework as available. Training samples are rejected if less than half of the volume contains ground truth annotation. As raw data and ground truth labels have different resolution in the quasi-isotropic prediction framework, nodes were extended to support augmentation of data with non-uniform voxel sizes as needed.

3.4.2.2 *Prediction & Reconstruction*

Affinity prediction on test data can be trivially parallelized over independent blocks in the output space that have the same size as the network output tensor. GPU idle time is minimized with the daisy framework[10] and the processing time per mega voxel and GPU is $t_p = 2.13\text{s} \pm 0.25\text{s}$. The increased resolution of the quasi-isotropic prediction allows for a reduction of the affinity vector at location p with N affinities into a scalar average

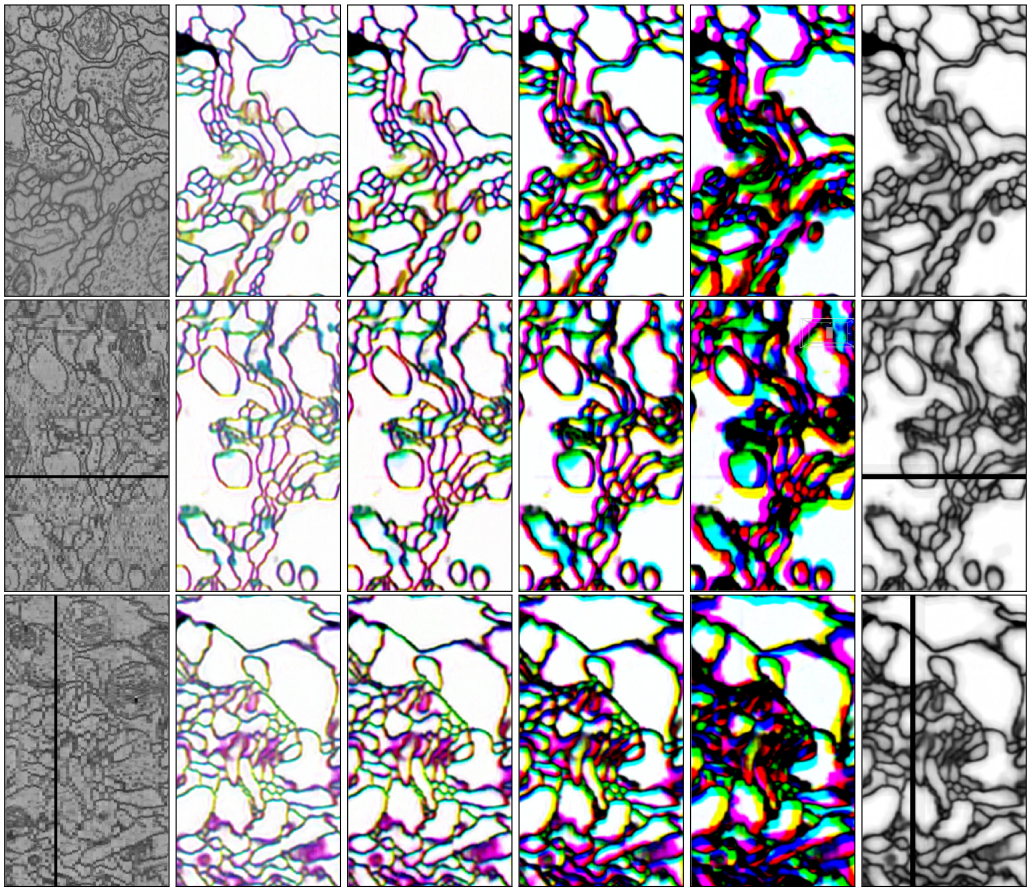
$$\bar{a}(p) = \frac{1}{N} \sum_{i=1}^{N} a_i(p) \tag{67}$$

without loss of boundary information to simplify the subsequent agglomeration task: Super-voxels can be extracted — and possibly agglomerated — from a scalar boundary map instead of the affinity graph, potentially with long-range affinities. Scaling the averaged affinities $\bar{a}(p)$ by the inverse of the glia predictions $1 - g_0^1(p)$ (section 3.4.1.1) suppresses potential incorrect high affinities within glia cells. Optionally, glia predictions above a user-specified threshold $t_g > 0$ are masked out and considered background during reconstruction. Figure 26 shows a small excerpt of affinity predictions for CREMI sample C.

Initial fragments are generated with a highly over-segmenting seeded watersheds transform on the square root of the distance transform (section 3.4.1.2) of the squared scaled affinity prediction average

$$D(p) = \sqrt{\min_{q \in \mathcal{G}} d(p,q) + \left(\left(1 - g_0^1(q)\right) \times \bar{a}(q)\right)^2} \tag{68}$$

[10] https://github.com/funkelab/daisy



(a) Raw (b) a_1, a_5, a_9 (c) a_2, a_6, a_{10} (d) a_3, a_7, a_{11} (e) a_4, a_8, a_{12} (f) $\bar{a}\,(1 - g_0^1)$

Figure 26: Excerpt of affinity predictions for CREMI sample C: The top, center, and bottom are screenshots of the top-left, top-right, and bottom-left cross-sectional views of Paintera, respectively. The screenshots were rotated to align better with the page layout. The raw data is displayed in column (a), followed by RGB-mapped subsets of the affinity predictions as indicated by the channel indices in (b)–(e), and the averaged affinities $\bar{a}$(f). The contrast range for the affinities was set to $[0.2, 1]$. Brighter pixels indicate high affinity, dark pixels indicate cell boundary or glia. White pixels in (b)–(e) have high affinity along all dimensions. While the resolution of the raw data is anisotropic, the original lower-resolution dimension cannot be identified in the quasi-isotropic predictions and the recovered level of details is remarkable.

with a weighted squared Euclidian distance

$$d(p,q) = a \sum_{i=1}^{3} w_i^2 \left(p_i - q_i\right)^2, \tag{69}$$

$$w = \begin{pmatrix} 1 \\ 1 \\ 10/9 \end{pmatrix}. \tag{70}$$

The weight vector w compensates for the slight an-isotropy in the quasi-isotropic predictions. The parameter a weighs the distance against the scaled affinity prediction average and can be considered a soft threshold. The distance transform helps close or narrow gaps in the boundary prediction and small initial fragments minimize the impact of under-segmentation. Smaller values of a result in wider boundaries. Watershed seeds are extracted as local maxima of the thresholded distance transform

$$D_t(p) = \begin{cases} D(p) & \text{if } D(p) > t_d, \\ -\infty & \text{otherwise.} \end{cases} \tag{71}$$

Local maxima below the threshold t_d are not considered seeds. Then, fragments are generated with seeded watersheds on the negative distance transform $-D(p)$. If $t_g < \infty$, watershed regions cannot grow into the thresholded glia prediction. Finally, in order to reduce the number of fragments drastically, any pair of fragments is merged if the median of affinities along their shared boundary is below a user-specified threshold t_m.

3.4.2.3 *Results*

Ultimately, the quality of neuron reconstruction from network predictions is the benchmark for the performance evaluation of the quasi-isotropic networks. This is true in particular for the affinity predictions and, thus, I evaluated the quasi-isotropic networks with respect to the quality of the subsequent neuron reconstruction instead of simply comparing voxel-wise affinity predictions to ground-truth. Additionally, I evaluated performance of the glial cell predictor by voxel-wise comparison of glial cell predictions with ground truth. The goal of the glial cell prediction is not the identification of individual glial cell instances but a semantic segmentation of voxels into glial and non-glial cells and, therefore, a voxel-wise evaluation is much more reasonable for this predictor. The evaluation of glial cell prediction for various settings for the threshold t_g also gives justification for the parameter range of t_g that was used in the grid search to determine the best performing network architecture with respect to neuron reconstruction. Figure 27 shows example network predictions for affinity and glial cells, neuron reconstruction, and ground truth on sample 1 for the best performing network architecture qi-mse-ng after 450,000 iterations of training with merge threshold $t_m = 0.7$ and glia threshold $t_g = 0.5$. Similar visualizations are provided for all samples in figures 59 to 63 of appendix B.1. Note that $t_g = 0.3$ would be a better choice (table 9) but table 22 shows that the difference is minuscule and evaluation shows that the glia threshold has only a minor effect on the segmentation outcome.

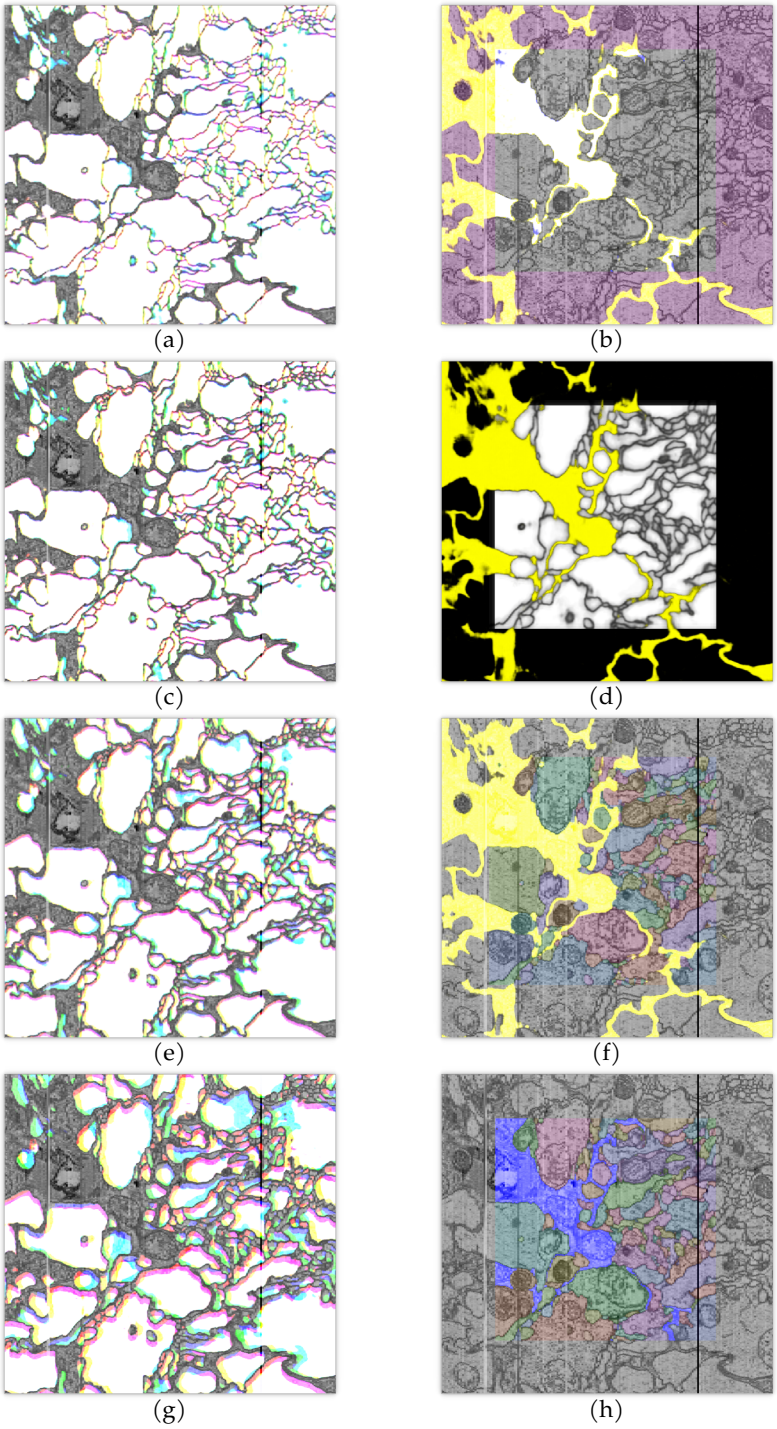


Figure 27: Cross-section of prediction and reconstruction for sample 1 with best performing network qi-mse-ng (table 9): The horizontal (x)/vertical (y) image axes are aligned with the third/second axis of the data. The corresponding resolution ($x \times y$)is 40nm × 4nm for raw data, and $^{40}/_{3}$nm × 12nm for interpolated ground truth, predictions and reconstruction. The annotated area spans 5×10^3 nm $= 5\,\mu$m from left to right (enclosed by magenta shading (b)). Each set of affinity channels, a_1, a_5, a_9(a), a_2, a_6, a_{10}(c), a_3, a_7, a_{11}(e), and a_4, a_8, a_{12}(g), is projected onto RGB channels with contrast range $[0, 1]$. Yellow glia predictions g with contrast range $[0, 1]$ are overlaid with blue ground truth in (b) to highlight false negative (blue), false positive (yellow), and correct predictions (white) within the masked area. Glia annotations are not available outside the annotated and yellow predictions cannot be considered false there. The averaged and scaled affinities $\bar{a}\left(1 - g_0^1\right)$ are restricted to the annotated volume (d). Predicted glia voxels are not considered for neuron reconstruction (f). The ground truth (h) is shown for comparison and with glia highlighted in blue color.

GLIAL CELL PREDICTION The bi-modal distribution of glia voxel predictions with two distinct peaks with similar mass as their respective ground truth counter-parts confirms robustness with respect to the glia threshold parameter t_g (figure 28): Varying the threshold t_g outside the two distinct peaks will only change a small number of predictions from positive to negative and vice versa. The histograms also show a high imbalance of labels — most voxels are not glia — and, thus, precision (what fraction of the positive predictions were correct) and recall (what fraction of all positive labels was recovered) are good metrics for evaluation:

$$\text{Precision} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fp}}, \tag{72}$$

$$\text{Recall} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fn}}, \tag{73}$$

where #tp is the number of true positives, *i.e.* correct classification as positive, #fp is the number of false positives, *i.e.* incorrect classification as positive, and #fn is the number of false negatives, *i.e.* incorrect classification as negative. The large number of true negative predictions is not considered and therefore does not skew precision or recall to give a false impression of good performance of classifiers that are good at predicting the over-represented negative class but fails to consistently predict the positive class. Precision-recall curves give a good intuition for the tradeoff between precision and recall. Figure 29 shows precision-recall curves for all networks and the highlighted precision-recall curve for the parameter range $0.1 \leq t_g \leq 0.9$ confirms robustness with respect to the glia threshold. Recall is high for all samples within the relevant parameter range with the exception of sample B. Similarly, precision is good for all samples with the exception of samples B and 2.

The F-measure

$$F_\beta = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{\beta^2 \text{precision} + \text{recall}} \tag{74}$$

with paramater $\beta \geq 0$ to balance between precision and recall combines precision-recall pairs into a single scalar value. For $\beta = 1$, the

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{75}$$

measure is the harmonic mean of precision and recall. The F_2 measure puts a stronger emphasis on the recall and, as high recall of glial cells reduces under-segmentation errors caused by confusion with glial cells, the plateaus of the F_2-measure in figure 30 for the interval $[0.1, 0.9]$ confirm that any specific choice of t_g within this interval should not have a strong effect on neuron reconstruction. The bad performance with respect to precision on samples B and 2, and in particular with respect to recall on sample B is reflected by lower F_2 scores across the entire parameter range.

AFFINITY PREDICTIONS Figure 31 shows examplary affinity prediction on the same subset of sample 1 for each network architecture. For visualization, affinities are

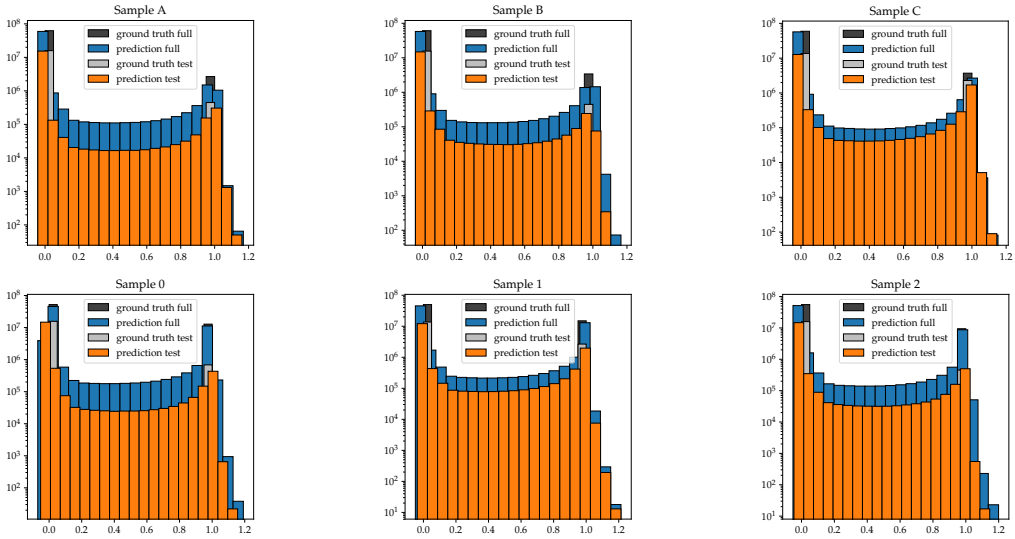
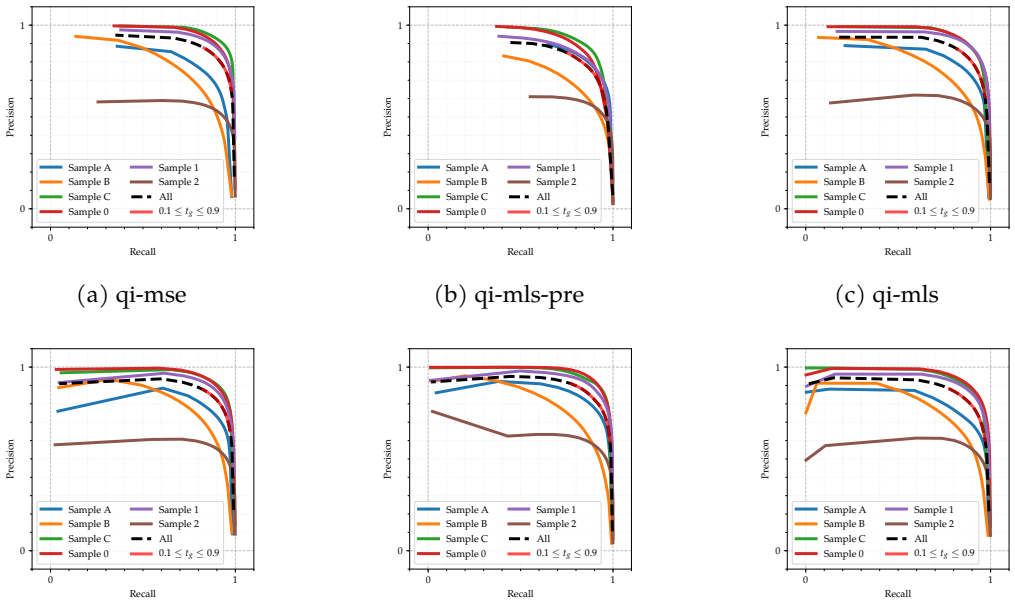


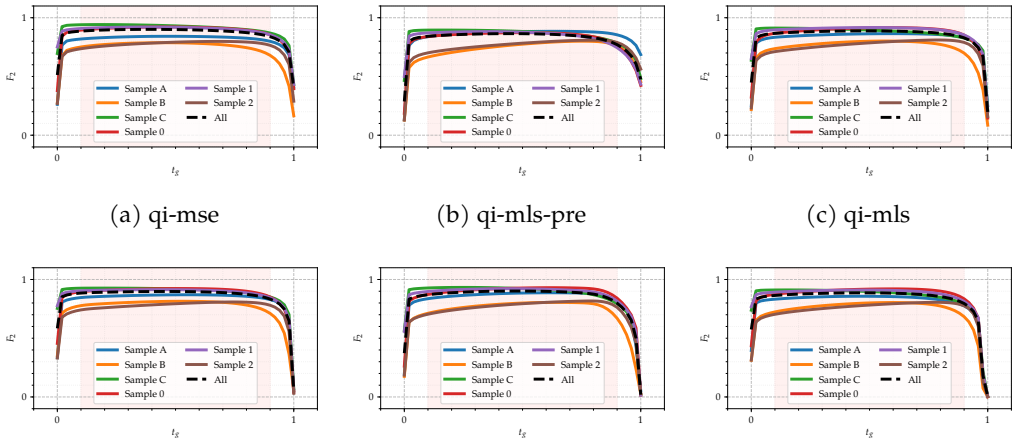
Figure 28: Histograms of glia voxel predictions for all samples A, B, C, 0, 1, 2 for network qi-mse-ng with log-scale y-axis. Histograms are provided for prediction on test set only or prediction on all annotated data with corresponding distributions of labels for ground truth annotations. The bi-modal distribution of predicted values with two distinct peaks suggests that the reconstruction is not sensitive for glia thresholded parameter $t_g \in [0.1, 0.9]$. Prediction histograms extend beyond $[0, 1]$ and have different binning than ground truth histograms because glia predictions are not restricted to the interval $[0, 1]$.



(a) qi-mse (b) qi-mls-pre (c) qi-mls

(d) qi-mse-ng (e) qi-mls-pre-ng (f) qi-mse-ng

Figure 29: Precision-recall curves for glia predictions for each network, evaluated per sample and combined for all samples. The parameter range $0.1 \leq t_g \leq 0.9$ used in the reconstruction experiments is highlighted on the combined precision-recall curve for all samples.



(a) qi-mse (b) qi-mls-pre (c) qi-mls

(d) qi-mse-ng (e) qi-mls-pre-ng (f) qi-mse-ng

Figure 30: F_2 measures for for glia predictions for each network, evaluated per sample and combined for all samples. The paramter range $0.1 \leq t_g \leq 0.9$ is highlighted as shaded area in each graph.

split into tuples of three channels that are then mapped into red, green, and blue channels and are overlaid over raw data. The subsets of channels are selected such that each channel triplet consists of one affinity channel for each of the three data dimensions and each affinity channel represents the same offset in units of voxels, *e.g.* (a_1, a_5, a_9). The affinities represented by the red, green, and blue color channels are aligned with the orthogonal of the image plane, the vertical image axis, and the horizontal image axis, respectively.

The predictions for the two mean-squared-error trained networks (indicated by blue shade), qi-mse and qi-mse-ng, do not differ qualitatively: The qi-mse network merely seems to predict slightly wider boundaries than the qi-mse-ng network. Smaller regions disappear for longer range-predictions as expected but still appear in the average $\bar{a}$.

For all malis trained networks (red shade), in contrast, many small objects are lost at the nearest-neighbor affinity predictions (a_1, a_5, a_9). Longer range affinity predictions do not produce any meaningful output at all — very small values around 0 or negative numbers — and, consequently, averaged affinities $\bar{a}$ are much smaller than expected: Averaged affinities are displayed with a contrast range of $[0.0, 0.3]$ for malis trained networks.

NEURON RECONSTRUCTION Figure 32 provides an intuition of topological correctness of reconstructed neurons with 3D mesh renderings of a randomly selected subset. Neuron topology appears biologically plausible and no severe under-segmentation errors can be observed. Examples of neuron reconstruciton results for various samples are provided in figure 33(c).

For each CREMI sample A, B, C, 0, 1, and 2 and each network architecture (table 3), I evaluated the performance for glia thresholds $t_g \in \{\infty\} \cup \{0.1, 0.2, \dots, 0.9\}$ and merge thresholds $t_m \in \{0.1, 0.3, \dots, 0.9\}$ with fixed distance transform weight $a = 0.1$ and distance transform threshold $t_d = 0.3$ for watersheed seed extrac-

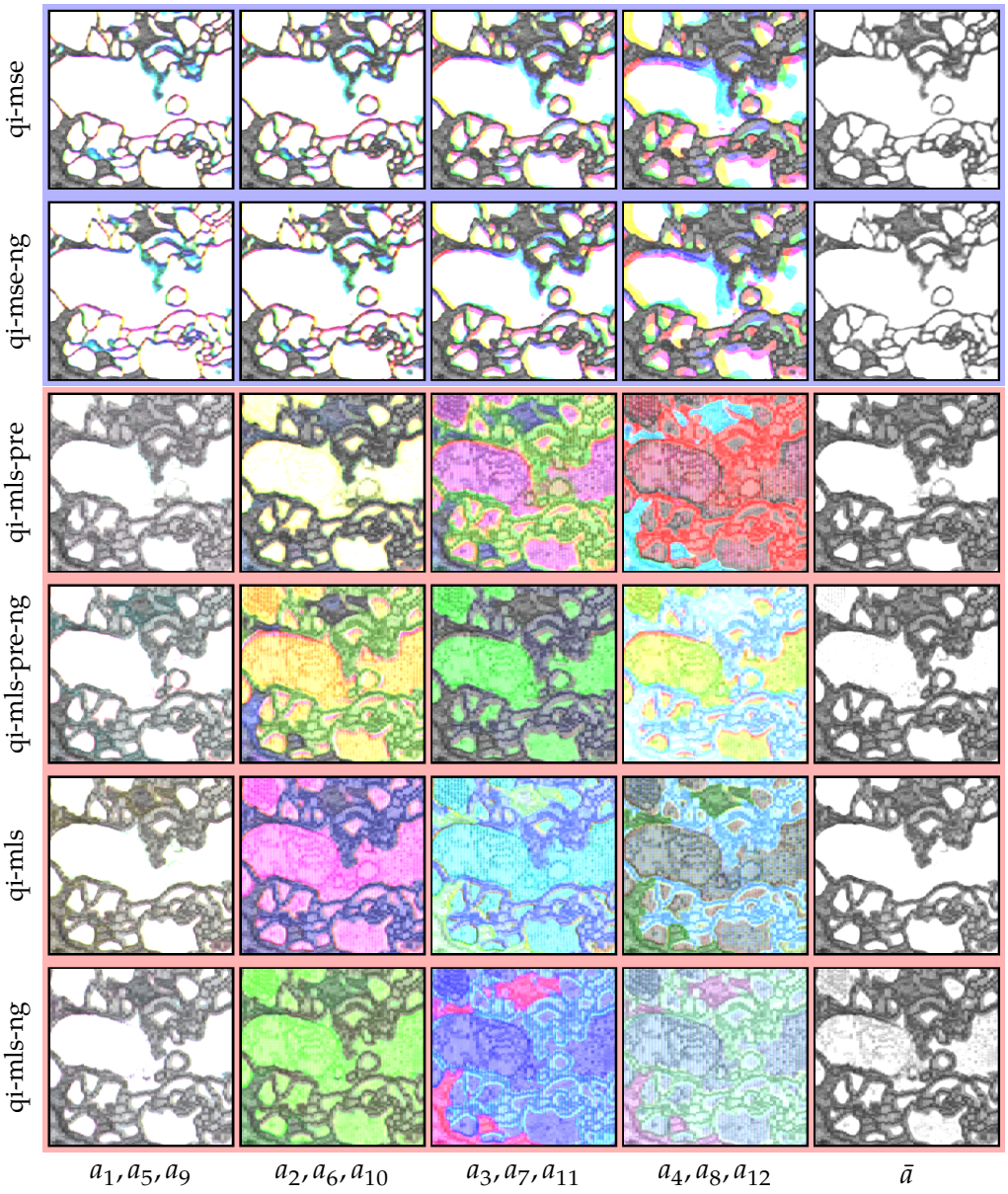


Figure 31: Examples of affinity predictions on sample 1 and average $\bar{a}$ for all networks. The low-resolution dimension of the raw data is aligned with the horizontal image axis. Predictions are split up into channels a_1, a_5, a_9 (first column), a_2, a_6, a_{10} (second), a_3, a_7, a_{11} (third), and a_4, a_8, a_{12} (fourth) that are mapped into RGB space with contrast ranges $[0.5, 0.1]$ for mean-squared-error trained networks (blue shade) and $[0.2, 1.0]$ for malis trained networks (red shade). Malis trained networks do not produce meaningful outputs for long-range affinities ($a_2, a_3, a_4, a_6, a_7, a_8, a_{10}, a_{11}, a_{12}$) and contrast ranges for those channels are chosen arbitrarily for visualization. The contrast ranges for the average affinities $\bar{a}$ are $[0.3, 1.0]$ and $[0.0, 0.3]$ for mean-squared-error trained networks and malis trained networks, respectively.

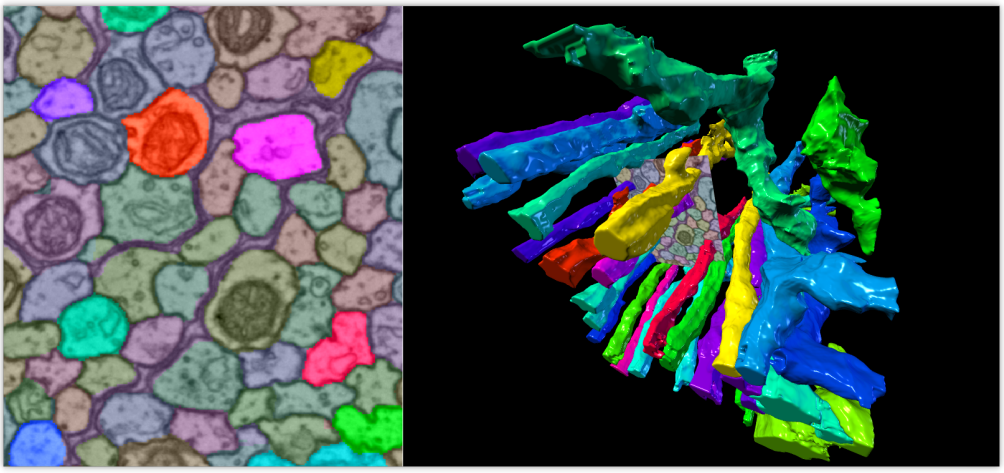


Figure 32: Neuron reconstrution for sample A with 3D rendering of randomly selected neurons. Affinities of network architecture qi-mse-ng were used with parameters $t_m = 0.7$ and $t_g = 0.5$.

tion. I report the CREMI metrics[11] *variation of information* for splits (VOI_s) and merges (VOI_m), the *adapted Rand error* (RAND), and the combined *CREMI score*

$$\text{CREMI} = \sqrt{(\text{VOI}_s + \text{VOI}_m) \times \text{RAND}}, \tag{76}$$

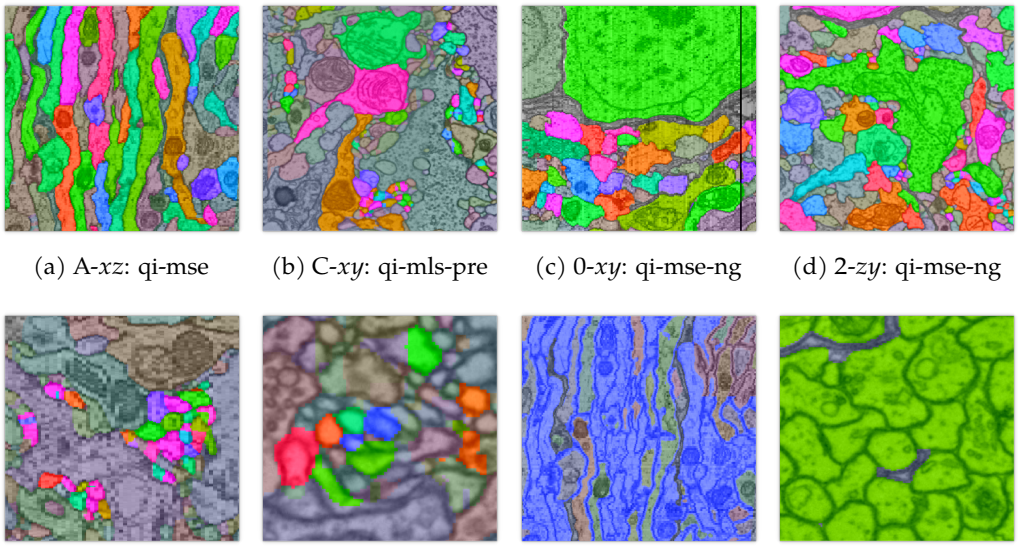
the geometric mean of the variation of information metrics and the adapted Rand error. For all of these metrics, a smaller value generally means better performance. Following the CREMI evaluation, I ignore ground truth voxels that are within 25nm of neuron boundary for evaluation to compensate for ambiguities of the neuron boundary that do not affect the overall reconstruction result. Contrary to the CREMI evaluation and due to the quasi-isotropic resolution of my predictions, I apply this boundary filter in three instead of two dimensions.

For each network and metric the best-performing parameters are those that minimize the average across all samples of (a) the CREMI score or (b) each individual metric. The metrics of the best-performing reconstruction parameter settings (t_m, t_g) of each networks are listed in tables 6 to 9 and visualized in figures 34 to 37. Networks are evaluated on either 25% of the data that were not used for training, or on the entire annotated data including the training data (table 5). The evaluation for the complete set of parameters is provided in tables 15 to 22 of appendix B.2.

Note that the CREMI leaderboard scores, when shown or listed for comparison, are averaged over performance on test samples A+, B+, C+, which are substantially larger than the 25% of the unused training data available for testing in my experiments. The average over all CREMI test samples is used because there are no CREMI evalutation data sets that are the equivalent of samples 0, 1, and 2 in my experiments. For better comparison, I show results on each individual sample A, B, C, 0, 1, and 2, as well as averages over all samples.

Consistent with the observations in the visual inspection, the mean-squared-error trained networks perform better overall: With the exception of qi-mls,

[11] https://cremi.org/metrics



(a) A-xz: qi-mse (b) C-xy: qi-mls-pre (c) 0-xy: qi-mse-ng (d) 2-zy: qi-mse-ng

(e) B-zy: qi-mse-ng (f) A-xz: qi-mls-pre (g) A-xz: qi-mls-pre (h) A-xy: qi-mls-ng

Figure 33: Quasi-isotropic neuron reconstruction examples and error modes for samples and network architectures as specified in captions. The axis alignment of horizontal and vertical image axes with the data is specified after the sample, *e.g.* xy. Large objects in sample A are reconstructed without obvious under-segmentation(a). Some neurons are highlighted for easier distinction of neighbors with similar color mapping. Both large and small neurons are recovered in sample C(b) but the large orange connected component contains false merges. Smaller objects are more likely to be over-segmented. Large objects in sample 0 are reconstructed without obvious under-segmentation(c). A mix of large and small objects in sample 2 is reconstructed in(d). Small objects are generally reconstructed heavily over-segmented(e) but under-segmentation can still occur(f). Reconstructions from malis-trained networks suffer from severe under-segmentation: Large parts (g) or almost all of the data(h) are merged into a single segment.

Table 5: The best performing parameter set (t_m, t_g) with respect to a metric is the parameter set that minimizes the average of either that metric or the CREMI score. Performance is evaluated on either the complete annotated data (100%) or on only the 25% that were not used for training. The respective tables and figures are listed in this table.

Data	Minimize	Table	Figure
25%	Individual Score	table 6	figure 34
100%	Individual Score	table 7	figure 35
25%	CREMI Score	table 8	figure 36
100%	CREMI Score	table 9	figure 37

Table 6: Performance of different network architectures evaluated on 25% of the data, excluding training data. Parameters are optimized for each metric individually. Score averages over all samples are listed in the ∅-column. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging. Metrics of the current CREMI leader by each individual metric are listed for comparison.

Metric	Architecture	t_m	t_g	∅	A	B	C	0	1	2
CREMI leader				0.318						
VOI_s	qi-mse	0.1	0.6	0.128	0.008	0.018	0.588	0.102	0.044	0.01
VOI_s	qi-mse-ng	0.1	0.6	0.117	0.006	0.011	0.62	0.003	0.056	0.007
VOI_s	qi-mls-pre	0.1	0.4	0.869	0.931	0.719	1.412	0.717	0.614	0.819
VOI_s	qi-mls-pre-ng	0.1	0.5	0.574	0.561	0.544	0.805	0.588	0.198	0.747
VOI_s	qi-mls	0.1	0.2	0.978	0.808	1.035	1.438	0.998	0.431	1.16
VOI_s	qi-mls-ng	0.1	0.1	0.078	0.0	0.0	0.0	0.148	0.053	0.266
CREMI leader				0.062						
VOI_m	qi-mse	0.9	0.9	0.074	0.011	0.171	0.038	0.107	0.048	0.072
VOI_m	qi-mse-ng	0.9	0.8	0.062	0.015	0.192	0.036	0.013	0.053	0.064
VOI_m	qi-mls-pre	0.5	0.9	0.966	1.068	1.151	0.835	1.035	0.517	1.189
VOI_m	qi-mls-pre-ng	0.3	0.9	2.148	2.39	2.035	1.952	2.174	1.624	2.711
VOI_m	qi-mls	0.5	0.9	0.081	0.033	0.112	0.084	0.06	0.067	0.128
VOI_m	qi-mls-ng	0.1	—	5.56	6.385	5.804	5.07	5.773	4.633	5.695
CREMI leader				0.108						
RAND	qi-mse	0.7	0.6	0.217	0.052	0.285	0.517	0.127	0.228	0.092
RAND	qi-mse-ng	0.7	0.3	0.23	0.085	0.302	0.52	0.127	0.242	0.105
RAND	qi-mls-pre	0.1	0.5	0.809	0.852	0.872	0.639	0.93	0.684	0.876
RAND	qi-mls-pre-ng	0.3	—	0.764	0.789	0.712	0.786	0.764	0.745	0.79
RAND	qi-mls	0.1	0.6	0.228	0.163	0.142	0.525	0.177	0.235	0.126
RAND	qi-mls-ng	0.1	—	0.99	1.0	1.0	1.0	0.979	0.999	0.963
CREMI leader				0.221						
CREMI	qi-mse	0.7	0.3	0.425	0.136	0.6	0.848	0.338	0.321	0.306
CREMI	qi-mse-ng	0.7	0.3	0.433	0.173	0.611	0.856	0.278	0.354	0.328
CREMI	qi-mls-pre	0.1	0.5	1.742	1.765	1.948	1.36	2.049	1.363	1.965
CREMI	qi-mls-pre-ng	0.3	0.4	1.803	1.834	1.755	1.834	1.789	1.673	1.933
CREMI	qi-mls	0.1	0.6	0.508	0.39	0.457	0.952	0.451	0.374	0.428
CREMI	qi-mls-ng	0.1	0.4	2.361	2.527	2.409	2.252	2.419	2.163	2.397

Malis trained architectures that show good performance with respect to under-segmentation fail with respect to over-segmentation, and vice versa. As a result, the overall CREMI score turns out higher (worse) for malis trained networks. In particular when comparing on the CREMI score optimizing parameters, all but one malis trained network (qi-mls-ng) perform badly with respect to the VOI_s and RAND metrics that consider over-segmentation. The comparably better performance of qi-mls-ng with respect to over-segmentation metrics can be easily understood in figure 33(h): Almost all voxels are grouped into a single connected component and, as a result, there is no over-segmentation. Accordingly, qi-mls-ng performs badly in all other metrics and is the worst network with respect to the CREMI score. The best-performing networks are the mean-squared-error trained networks qi-mse and qi-mse-ng, closely followed only by one malis trained network, qi-mls.

I evaluated the effect of the merge threshold t_m for a fixed value of the glia threshold t_g (figure 38), and vice versa (figure 39), for the best performing architecture qi-mse-ng. In accordance with table 9 I choose $t_m = 0.7$ and $t_g = 0.3$, respectively. Small values of t_m achieve good performance on the over-segmentation metric VOI_s but perform badly on the under-segmentation metric VOI_m.

Larger values of t_m improve performance on VOI_m at the expense of worse VOI_s scores, but the effect is not as dramatic, which is reflected in the combined RAND and CREMI scores: Even a heavily over-segmenting merge threshold $t_m = 0.9$ performs better than a seemingly moderate choice for $t_m = 0.5$ and smaller. The ideal merge threshold is $t_m = 0.7$, which is in agreement with table 9 disregarding networks trained with malis loss (figure 31).

Table 7: Performance of different network architectures evaluated on 100% of the data, including training data. Parameters are optimized for each metric individually. Score averages over all samples are listed in the ∅-column. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging. Metrics of the current CREMI leader by each individual metric are listed for comparison.

Metric	Architecture	t_m	t_g	∅	A	B	C	0	1	2
CREMI leader				0.318						
VOI_s	qi-mse	0.1	0.6	0.144	0.004	0.026	0.763	0.042	0.018	0.012
VOI_s	qi-mse-ng	0.1	0.6	0.14	0.002	0.025	0.772	0.006	0.022	0.01
VOI_s	qi-mls-pre	0.1	0.6	0.803	0.859	0.657	1.611	0.441	0.75	0.503
VOI_s	qi-mls-pre-ng	0.1	0.2	0.549	0.423	0.536	0.765	0.56	0.519	0.49
VOI_s	qi-mls	0.1	0.3	0.984	0.956	1.062	2.04	0.377	0.711	0.756
VOI_s	qi-mls-ng	0.1	0.1	0.141	0.377	0.068	0.13	0.087	0.051	0.132
CREMI leader				0.062						
VOI_m	qi-mse	0.9	0.9	0.031	0.005	0.066	0.013	0.044	0.021	0.039
VOI_m	qi-mse-ng	0.9	0.8	0.03	0.012	0.082	0.013	0.005	0.031	0.036
VOI_m	qi-mls-pre	0.5	0.9	0.883	1.007	1.097	1.171	0.446	0.752	0.825
VOI_m	qi-mls-pre-ng	0.5	0.9	2.127	2.371	1.86	2.704	1.7	1.765	2.362
VOI_m	qi-mls	0.5	0.9	0.067	0.027	0.085	0.093	0.024	0.084	0.088
VOI_m	qi-mls-ng	0.3	—	6.014	7.343	6.528	6.361	4.598	5.554	5.703
CREMI leader				0.108						
RAND	qi-mse	0.7	0.6	0.356	0.167	0.16	0.398	0.222	0.732	0.456
RAND	qi-mse-ng	0.7	0.8	0.36	0.165	0.18	0.403	0.224	0.733	0.455
RAND	qi-mls-pre	0.1	0.6	0.836	0.971	0.928	0.807	0.496	0.908	0.907
RAND	qi-mls-pre-ng	0.3	—	0.845	0.817	0.784	0.841	0.917	0.871	0.841
RAND	qi-mls	0.1	—	0.387	0.25	0.141	0.422	0.41	0.642	0.457
RAND	qi-mls-ng	0.3	—	0.988	0.966	0.992	0.997	0.998	0.998	0.979
CREMI leader				0.221						
CREMI	qi-mse	0.7	0.5	0.522	0.312	0.423	0.873	0.28	0.693	0.551
CREMI	qi-mse-ng	0.7	0.3	0.519	0.288	0.485	0.871	0.243	0.689	0.539
CREMI	qi-mls-pre	0.1	0.6	1.832	2.178	2.117	1.903	1.083	1.819	1.894
CREMI	qi-mls-pre-ng	0.3	0.4	1.954	1.962	1.911	2.156	1.798	1.874	2.025
CREMI	qi-mls	0.1	0.4	0.647	0.505	0.424	1.004	0.464	0.82	0.667
CREMI	qi-mls-ng	0.3	0.7	2.466	2.727	2.558	2.544	2.174	2.367	2.424

Table 8: Performance of different network architectures evaluated on 25% of the data, excluding training data. Parameters that minimize CREMI score are used for all metrics. Score averages over all samples are listed in the ∅-column. Metrics of the current CREMI leader by overall CREMI score are listed for comparison.

Metric	Architecture	t_m	t_g	∅	A	B	C	0	1	2
CREMI leader				0.339						
VOI_s	qi-mse	0.7	0.3	0.797	0.564	0.878	1.841	0.298	0.591	0.611
VOI_s	qi-mse-ng	0.7	0.3	0.722	0.458	0.778	1.768	0.212	0.554	0.563
VOI_s	qi-mls-pre	0.1	0.5	0.825	0.86	0.658	1.608	0.599	0.731	0.496
VOI_s	qi-mls-pre-ng	0.3	0.4	2.216	2.392	2.733	2.748	1.488	1.857	2.082
VOI_s	qi-mls	0.1	0.6	0.992	0.962	1.067	2.046	0.378	0.74	0.757
VOI_s	qi-mls-ng	0.1	0.4	0.153	0.377	0.068	0.13	0.116	0.052	0.174
CREMI leader				0.115						
VOI_m	qi-mse	0.7	0.3	0.087	0.02	0.255	0.071	0.058	0.059	0.057
VOI_m	qi-mse-ng	0.7	0.3	0.131	0.059	0.391	0.115	0.052	0.094	0.074
VOI_m	qi-mls-pre	0.1	0.5	3.221	4.019	4.177	2.876	1.919	2.881	3.452
VOI_m	qi-mls-pre-ng	0.3	0.4	2.184	2.373	1.88	2.707	1.859	1.842	2.442
VOI_m	qi-mls	0.1	0.6	0.223	0.091	0.288	0.304	0.269	0.193	0.193
VOI_m	qi-mls-ng	0.1	0.4	6.031	7.386	6.528	6.365	4.61	5.551	5.745
CREMI leader				0.108						
RAND	qi-mse	0.7	0.3	0.356	0.167	0.163	0.398	0.222	0.732	0.456
RAND	qi-mse-ng	0.7	0.3	0.363	0.16	0.201	0.403	0.224	0.732	0.455
RAND	qi-mls-pre	0.1	0.5	0.853	0.97	0.928	0.807	0.595	0.907	0.907
RAND	qi-mls-pre-ng	0.3	0.4	0.879	0.807	0.792	0.852	0.966	0.95	0.907
RAND	qi-mls	0.1	0.6	0.389	0.242	0.131	0.428	0.326	0.738	0.466
RAND	qi-mls-ng	0.1	0.4	0.992	0.969	0.992	0.997	0.999	1.0	0.994
CREMI leader				0.221						
CREMI	qi-mse	0.7	0.3	0.523	0.312	0.429	0.872	0.281	0.69	0.552
CREMI	qi-mse-ng	0.7	0.3	0.519	0.288	0.485	0.871	0.243	0.689	0.539
CREMI	qi-mls-pre	0.1	0.5	1.854	2.176	2.118	1.903	1.224	1.811	1.892
CREMI	qi-mls-pre-ng	0.3	0.4	1.954	1.962	1.911	2.156	1.798	1.874	2.025
CREMI	qi-mls	0.1	0.6	0.647	0.504	0.422	1.003	0.46	0.83	0.665
CREMI	qi-mls-ng	0.1	0.4	2.469	2.743	2.558	2.544	2.173	2.367	2.426

Table 9: Performance of different network architectures evaluated on 100% of the data, including training data. Parameters that minimize CREMI score are used for all metrics. Score averages over all samples are listed in the ∅-column. Metrics of the current CREMI leader by overall CREMI score are listed for comparison.

Metric	Architecture	t_m	t_g	∅	A	B	C	0	1	2
CREMI leader				0.339						
VOI_s	qi-mse	0.7	0.5	0.803	0.565	0.884	1.849	0.301	0.606	0.614
VOI_s	qi-mse-ng	0.7	0.3	0.722	0.458	0.778	1.768	0.212	0.554	0.563
VOI_s	qi-mls-pre	0.1	0.6	0.803	0.859	0.657	1.611	0.441	0.75	0.503
VOI_s	qi-mls-pre-ng	0.3	0.4	2.216	2.392	2.733	2.748	1.488	1.857	2.082
VOI_s	qi-mls	0.1	0.4	0.985	0.962	1.061	2.037	0.38	0.716	0.753
VOI_s	qi-mls-ng	0.3	0.7	0.159	0.381	0.068	0.13	0.128	0.053	0.194
CREMI leader				0.115						
VOI_m	qi-mse	0.7	0.5	0.079	0.018	0.235	0.067	0.052	0.051	0.051
VOI_m	qi-mse-ng	0.7	0.3	0.131	0.059	0.391	0.115	0.052	0.094	0.074
VOI_m	qi-mls-pre	0.1	0.6	3.225	4.03	4.172	2.875	1.924	2.894	3.452
VOI_m	qi-mls-pre-ng	0.3	0.4	2.184	2.373	1.88	2.707	1.859	1.842	2.442
VOI_m	qi-mls	0.1	0.4	0.231	0.092	0.302	0.314	0.277	0.198	0.201
VOI_m	qi-mls-ng	0.3	0.7	6.015	7.323	6.528	6.365	4.603	5.552	5.718
CREMI leader				0.108						
RAND	qi-mse	0.7	0.5	0.356	0.167	0.16	0.398	0.222	0.732	0.456
RAND	qi-mse-ng	0.7	0.3	0.363	0.16	0.201	0.403	0.224	0.732	0.455
RAND	qi-mls-pre	0.1	0.6	0.836	0.971	0.928	0.807	0.496	0.908	0.907
RAND	qi-mls-pre-ng	0.3	0.4	0.879	0.807	0.792	0.852	0.966	0.95	0.907
RAND	qi-mls	0.1	0.4	0.388	0.242	0.132	0.429	0.327	0.736	0.466
RAND	qi-mls-ng	0.3	0.7	0.991	0.965	0.992	0.997	0.999	1.0	0.994
CREMI leader				0.221						
CREMI	qi-mse	0.7	0.5	0.522	0.312	0.423	0.873	0.28	0.693	0.551
CREMI	qi-mse-ng	0.7	0.3	0.519	0.288	0.485	0.871	0.243	0.689	0.539
CREMI	qi-mls-pre	0.1	0.6	1.832	2.178	2.117	1.903	1.083	1.819	1.894
CREMI	qi-mls-pre-ng	0.3	0.4	1.954	1.962	1.911	2.156	1.798	1.874	2.025
CREMI	qi-mls	0.1	0.4	0.647	0.505	0.424	1.004	0.464	0.82	0.667
CREMI	qi-mls-ng	0.3	0.7	2.466	2.727	2.558	2.544	2.174	2.367	2.424

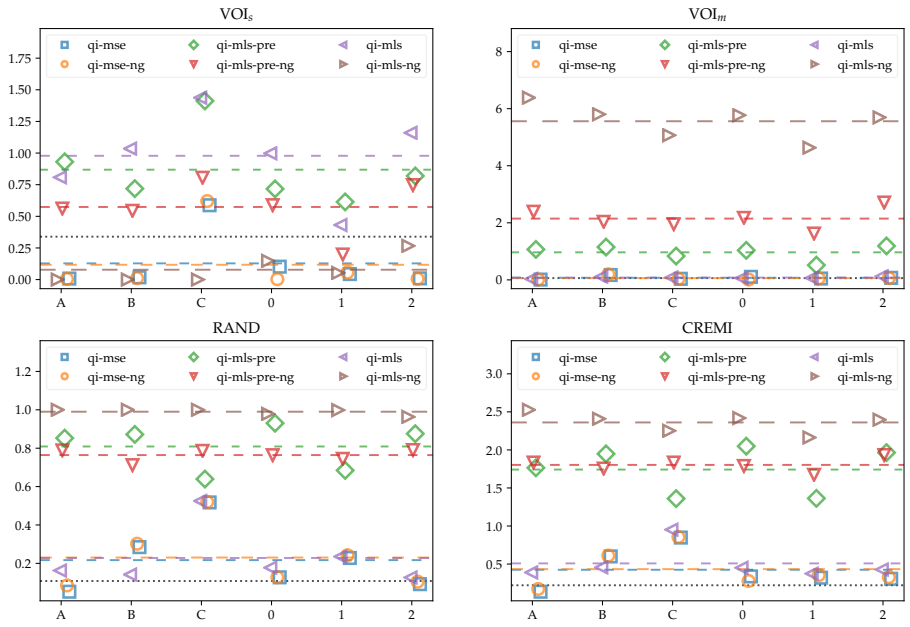


Figure 34: Performance of different network architectures evaluated on 25% of the data, including training data. Parameters are optimized for each metric individually. Score averages over all samples are indicated by horizontal dashes lines in matching colors. The current best scores as reported on the CREMI web site is shown as a black dotted horizontal line.

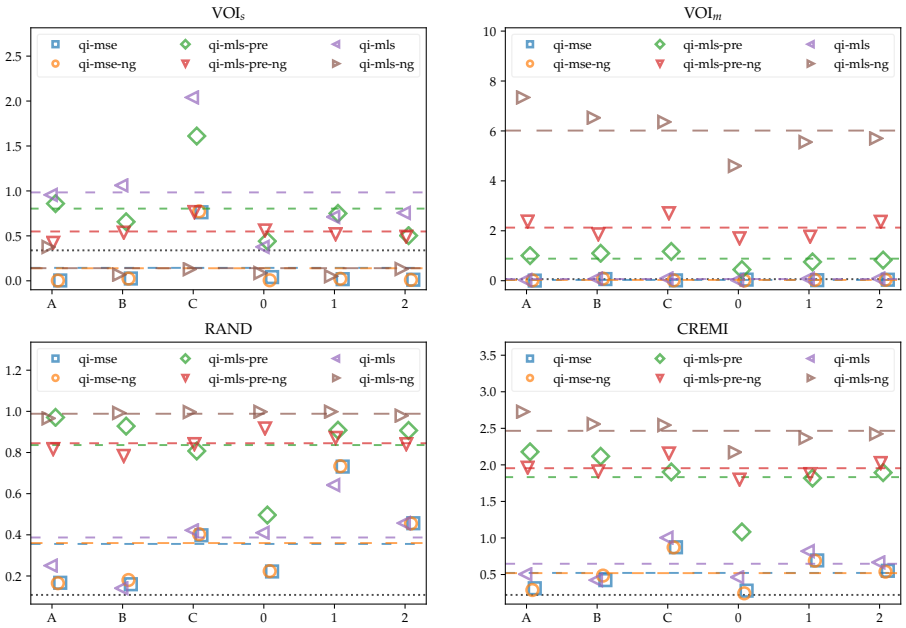


Figure 35: Performance of different network architectures evaluated on 100% of the data, including training data. Parameters are optimized for each metric individually. Score averages over all samples are indicated by horizontal dashed lines in matching colors. The current best score as reported on the CREMI web site is shown as a black dotted horizontal line.

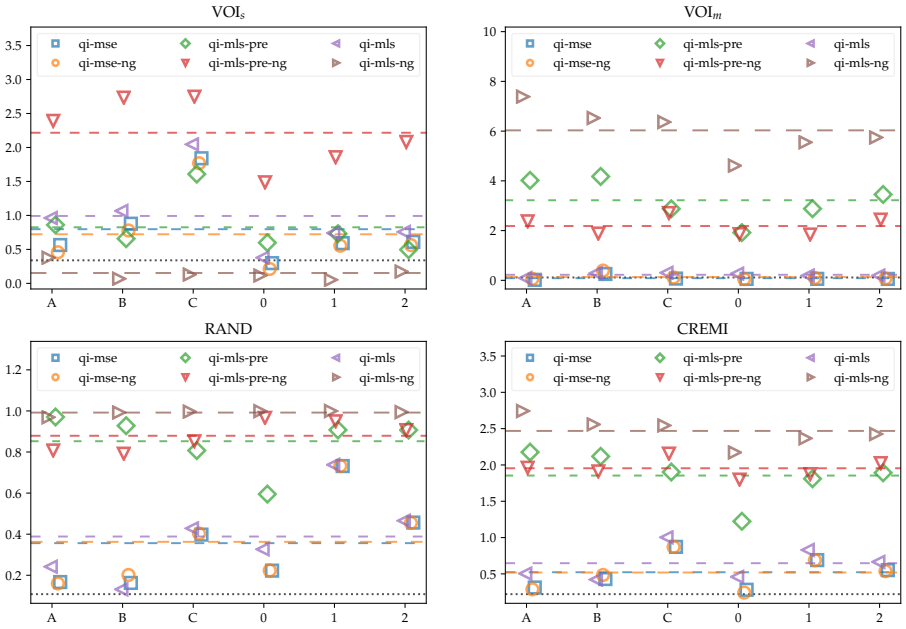


Figure 36: Performance of different network architectures evaluated on 25% of the data, including training data. Parameters that minimize CREMI score are used for all metrics. Score averages over all samples are indicated by horizontal dashed lines in matching colors. The current best score as reported on the CREMI web site is shown as a black dotted horizontal line.

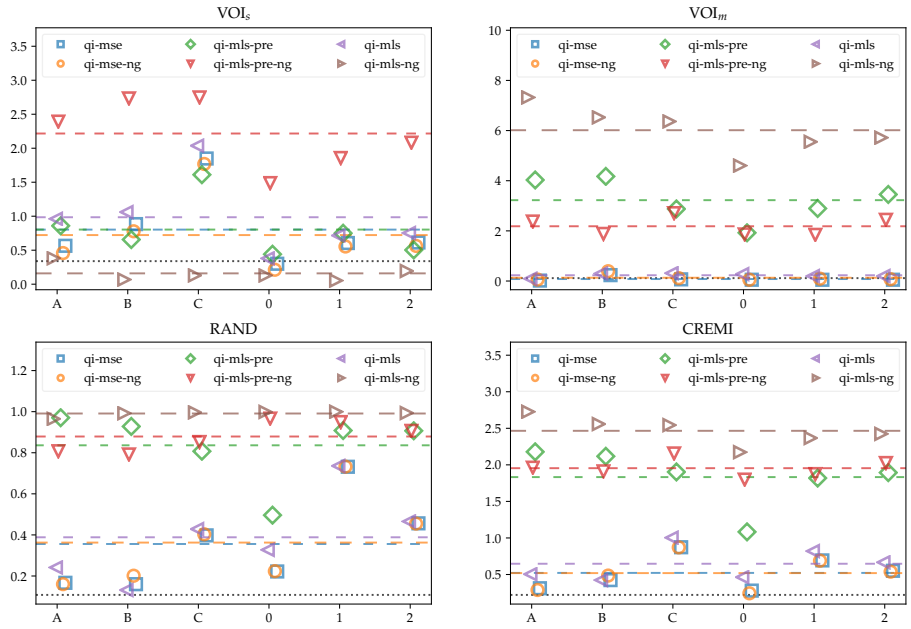


Figure 37: Performance of different network architectures evaluated on 100% of the data, including training data. Parameters that minimize CREMI score are used for all metrics. Score averages over all samples are indicated by horizontal dashed lines in matching colors. The current best score as reported on the CREMI web site is shown as a black dotted horizontal line.

The particular choice of t_g on the other hand does not have a strong effect on performance with respect to any of the metrics. Only disregarding the glia prediction for fragment extraction and merging seems to negatively affect the reconstruction, resulting in significantly worse performance. Even for a glia threshold as small as $t_g = 0.1$ the impact on the score appears to be insiginficant, which is consistent with the evaluation of glia prediction performance for various thresholds t_g in figures 29 and 30 and the very distinctly bi-modal distribution of glia predictions (figure 28). Performance varies by sample: In particular the poor performance on sample B with respect to the under-segmentation metric VOI_m is remarkable. The low recall of glial cells on sample B (figures 29 and 30) indicates that glial cell related under-segmentation is a major contributing factor.

3.4.3 *Discussion*

I created a novel convolutional neural network architecture to predict quasi-isotropic 3D affinity graphs from anisotropic raw data with an additional supplementary learning task of identifying glial cells. The supplementary predictions can be used to minimize under-segmentation error modes. I evaluated six different network architectures and showed that, with the right loss, good performance can be achieved with respect to under-segmentation metrics. The slightly worse performance with respect to over-segmentation metrics indicates that the best network architectures and parameters tend to over-segment. In general, the

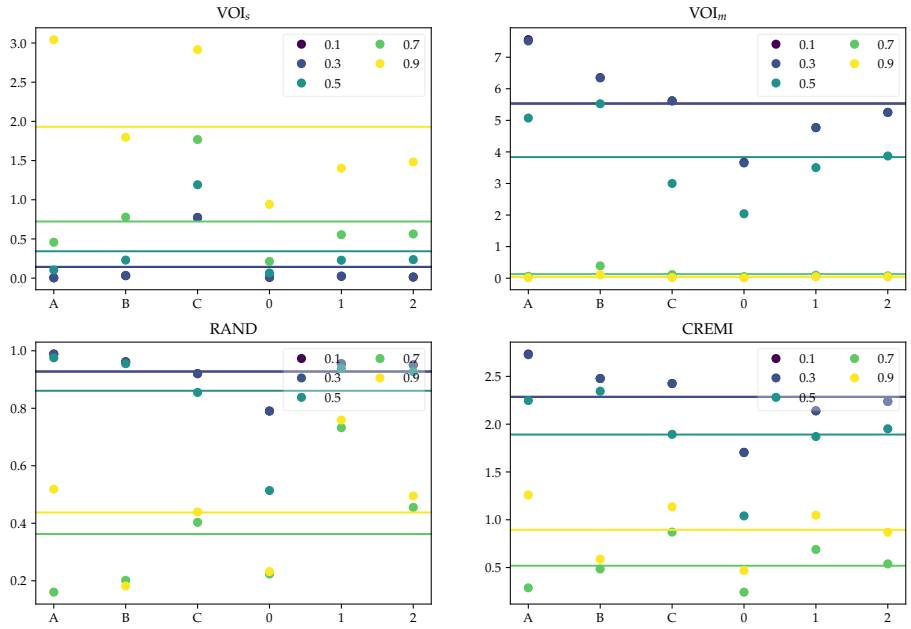


Figure 38: Evaluation of merge threshold for a fixed glia threshold $t_g = 0.3$ on 100% of the data for network qi-mse-ng. Merge thresholds $t_m \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ are represented by color-coded markers The performance for each merge threshold is plotted per sample. The average over all samples is displayed as a horizontal line in the same color. As expected, lower thresholds score better with respect to VOI$_s$, higher thresholds perform better overall and in particular with respect to VOI$_m$.

over-segmentation error mode is preferred over under-segmentation as it can be corrected for more rapidly in subsequent proof-reading sessions.

Currently, two watershed fragments are merged greedily if the median of all affinities along their shared boundary is above a user-specified threshold t_m. In order to avoid under-segmentation, t_m has to be set very conservatively. The resulting over-segmentation could be further reduced with more advanced agglomeration schemes like multicuts that solve a globally optimal configuration of merged and split edges, or hierarchical agglomeration with adaptive edge weights that are adjusted whenever two fragments are merged.

Additionally, while the median counteracts the effect to some extent, using the predicted affinities as merge criterion propagates bad performance of the network to the merging stage. Instead, a richer set of both fragment and boundary features on the predicted affinities as well as on the actual raw data could improve agglomeration, in particular when combined into a scalar score with a machine learning classifier such as random forests (Breiman 2001). Such features include but are not limited to size, statistical moments of the raw data or affinities with respect to fragments and boundaries, ratios of moments of fragments or fragments and boundaries, histograms of raw data or affinities, and shape features of fragments and boundaries. Preferably, features should be used that can be easily combined, *e.g.* additive features like histograms, in particular when applying a hierarchical agglomeration scheme or when distributed computation of the features is of the essence.

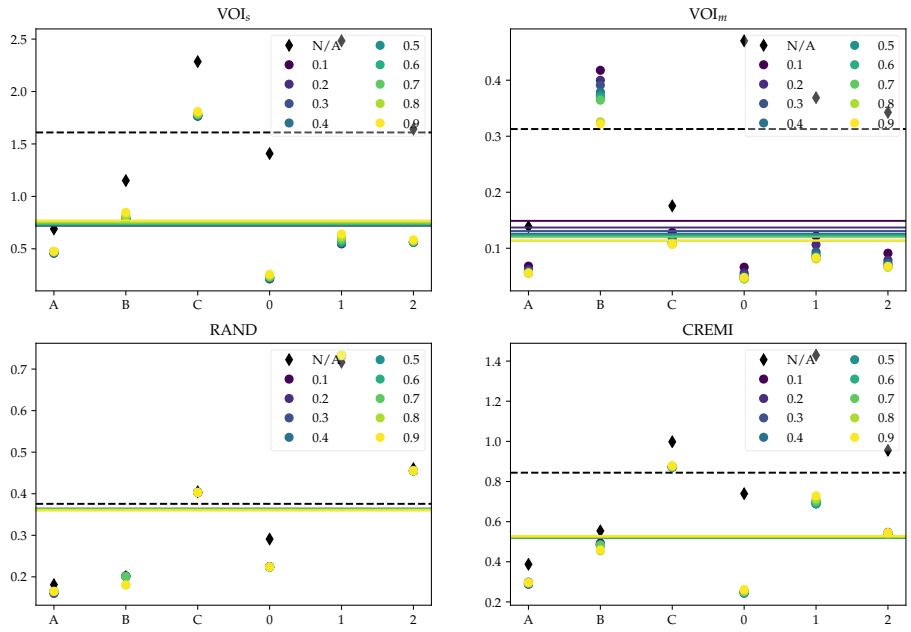


Figure 39: Evaluation of glia threshold for a fixed merge threshold $t_m = 0.7$ on 100% of the data. Glia thresholds $t_g \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ are represented by color-coded markers. The average over all samples is displayed as a horizontal line in the same color. Black diamond markers and a dashed black line indicate the performance when glia mask is not considered during reconstruction. Considering the glia during reconstruction generally performs better. The effect of the particular choice of t_g within the evaluated range is negligible compared to t_m (figure 38).

The major benefit of quasi-isotropic predictions is the extraction of 3D fragments (super-voxels) instead of 2D fragments as is common in neuron reconstruction from anisotropic data. As a result, fragments tend to be larger — and, therefore, the total number of fragments is smaller — and thus agglomeration becomes more effficient or in some cases (expensive agglomeration schemes and large datasets) feasible at all.

Another benefit is the recovery of small processes at about the size of the lowest spatial resolution, *e.g.* so-called "nests" that are hard to trace even for expert proof-readers. Reasonable segmentation hypotheses in the form of over-segmenting fragments are expected to benefit proof-reading: experts can rely on the fragments as guidance and only need to agglomerate them into a meaningful segmentation.

Surprisingly, all malis trained networks performed badly in the neuron reconstruction evaluation. The explanation can be found in visualizations of the affinity predictions (figure 31): While nearest-neighbor affinity predictions appear to be reasonable, predicted longer range affinities do not, but rather seem to have been barely trained at all. Indeed, with long range affinities, redundancy is added to the affinity graph and it is possible that the malis loss ignores those redundant paths. This may be resolved by summing separate losses for each of the affinity triplets (a_1, a_5, a_9), (a_2, a_6, a_{10}), (a_3, a_7, a_{11}), and (a_4, a_8, a_{12}) instead of a single loss for $(a_1, a_2, \dots, a_{12})$. However, already for the nearest-neighbor affinities, the visualizations show that some small objects are not recovered and it is questionable if networks trained with a separate loss per each affinity triplet would outperform mean-squared-error trained networks.

It may seem counter-intuitive that the performance on 25% of the data not used during training is better than on the entire dataset, 75% of which was used for training. Assuming a constant error-probability per cable length and that biological neuron cable length is proportional to the size of the observed dataset, it becomes clear that, in general, errors are more likely for larger datasets because the cable length of individual neurons tends to be larger. This is true in particular for the more catastrophic under-segmentation or merge error mode. It will thus be crucial not only for this network architecture to evaluate performance on even larger datasets than what is available in the CREMI challenge.

Label data interpolation is not a trivial task and the current approach of interpolating signed distance transforms tends to shrink objects (section 3.4.1.3). In the extreme case of no overlap, objects can vanish in the interpolated sections (figure 24). Most aggloemration schemes fail to connect objects without shared boundaries and quasi-isotropic reconstructions that were trained with ground truth generated by this interpolation scheme tend to suffer more from over-segmentation.

Future work will focus on creating better quasi-isotropic ground truth data by improving the label interpolation and the application of advanced agglomeration schemes like hierarchical agglomeration or multi-cut, possibly in combination with on-line learning concepts as discussed in section 4.1.8. The lowest-hanging fruit for improving label interpolation is aligning object masks by their respective centers of mass before interpolation as suggested in figure 25. In the presence

of multiple disconnected and possibly non-convex shapes for the same object id within a single section, it will be necessary to describe shapes more specifically and match shapes across sections. Polygons with a fixed number of vertices, for example, are a good candidate because they can be trivially interpolated through their individual vertices.

The significant variability of section thickness in TEM (chapter 2) should be considered during label interpolation for truly quasi-isotropic ground truth: Currently, labeled sections are positioned along the axial dimension at the equi-distant centers of the raw data sections without considering axial distortion. Instead, the labeled sections should be centered at the estimated locations before interpolation which may not be equi-distant or at the centers of the raw sections. The spacing of sections in the interpolated target space would remain the same: three sections at equi-distant axial spacing per raw section.

Another lane of future research is the improvement of local shape descriptors for EM neuron segmentation. Current work in metric learning explores the possibility of learning feature vectors that are similar within neurons and dissimilar across neuron boundaries for segmentation by clustering (Luther and Seung 2019). Like affinity graphs, in particular in the presence of long range affinities, those feature vectors can be interpreted as a local shape descriptor but the shape description is only implicit. Schmidt et al. (2018) use star-convex polygons to explicitly describe the shape of cells and nuclei for detection and segmentation. This nucleus detector could be adapted for EM neuron segmentation as a local shape descriptor: Instead of describing a complete nucleus, the star-convex polygon extends to the nearest boundary from the center voxel along predefined radial directions. All local shape predictors then cast votes to create a boundary map. I expect neuron reconstruction to benefit from star-shaped polygons because they are more flexible in describing local shape than affinity graphs. While affinity graphs can only detect boundaries at pre-defined distances along pre-defined directions, star-shaped polygons can detect boundaries at any distance along pre-defined directions at equi-distant angles. Star-convex polygons are less expressive in highly anisotropic TEM because many of the bounded rays along these directions pass through the same set voxels and should be learned and predicted with a quasi-isotropic architecture, as well.

4

SOFTWARE

During the course of my PhD, I created or contributed to various software libraries. In this chapter, I present a selection of my most important work. First, I developed *Paintera* (section 4.1), an extendable visualization tool for the rapid and efficient generation of densely annotated ground-truth and proof-reading of large 3D-EM connectomics. The success of deep learning models in neuron reconstruction relies on the availability of large amounts of high quality densely annotated ground truth data. Attempts to generate ground-truth annotations for the CREMI[1] challenge with existing tools failed to achieve a satisfying level of correctness. Consequently, we developed BigCAT as a targeted solution for realitively small datasets[2] to improve the CREMI ground truth annotations. I developed Paintera as the direct successor of BigCAT and re-used the annotation and proof-reading functionality that I had initially developed for BigCAT.

My extensive experiments with neuron reconstruction at quasi-isotropic resolution from anisotropic 3D-EM (chapter 3) required a reliable framework for the creation of reproducible experiments. I created the *EQIP* library (section 4.2) to create, run, and monitor these experiments.

During my time as a scientist I have had experience with two of the most popular ecosystems for multi-dimensional image processing and analysis in bioinformatics: The multi-dimensional array of the NumPy library is the fundamental data structure of many image processing and analysis frameworks and has been a major contributor to the success of Python programming language in scientifict computing. ImageJ is an established Java framework for the processing and analysis of multi-dimensional biological images. The ImageJ distribution Fiji provides a wealth of tools that are not available in other frameworks. Both Python with NumPy and ImageJ are extremely popular for bioinformatics research. Java programs are executed in a virtual machine and cannot access native memory programs like NumPy arrays. As a result, simultaneous use of NumPy and ImageJ in a single workflow is extremely cumbersome at best. I developed the *imglyb* Python library (section 4.3) to bridge the chasm between NumPy based image processing and analysis frameworks and modern ImageJ2 that builds upon the multi-dimensional generic image processing Java library ImgLib2.

One of the major benefits of Java over native languages is the ease of distribution: Java runs in a JVM and software can be distributed without platform-specific compilation. Build automation tools like Apache Maven[3] have simplified the build management and deployment process through online repositories but there is no straight-forward way to execute a Java application from Maven coordinates alone.

[1] https://cremi.org

[2] CREMI still provides the larges amount of ground truth annotations for 3D-EM connectomics to date.

[3] https://maven.apache.org

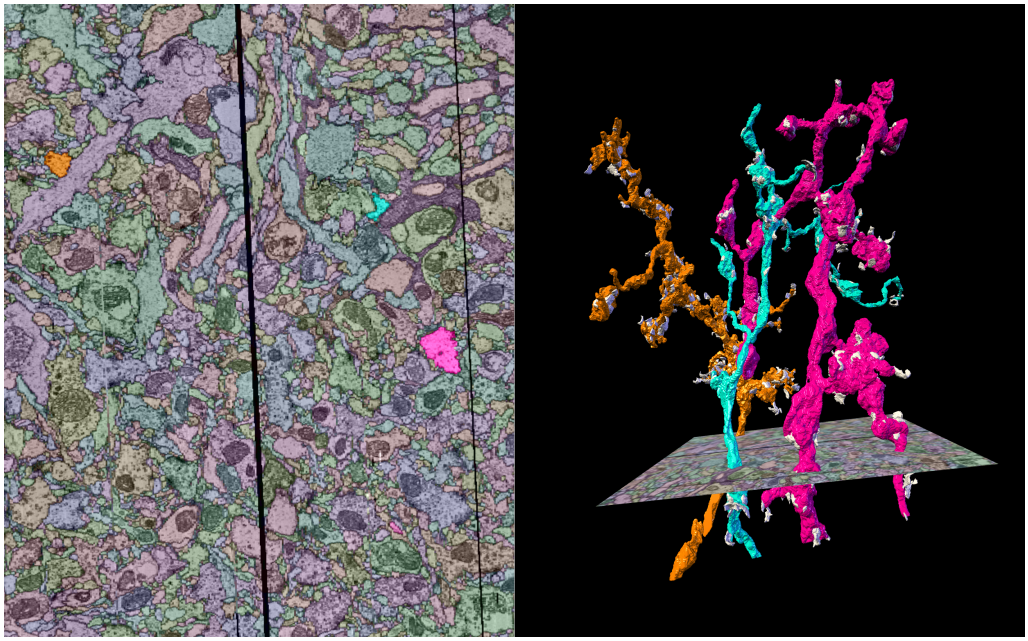


Figure 40: Chris Patrick captured this screenshot of Paintera while proof-reading the automatic reconstruction by Constantin Pape (Beier et al. 2017) of a small subset of the FAFB dataset (Zheng et al. 2018). The left view shows a 2D cross-section at abitrary rotation. Selected neurons are highlighted with higher opacity and visualized in 3D as triangle meshes on the right. Synaptic cleft predictions by Larissa Heinrich (Heinrich, Funke, et al. 2018) that intersect with selected neurons are rendered as white triangle meshes (section 4.1.5.4). All triangle meshes are generated on demand and memory-cached.

The jgo[4] command line tool executes Java applications from Maven coordinates. I developed a Python interface for *jgo* (section 4.4) that simplifies distribution of Java software: Any application that is available as a Maven artifact can now be distributed through the Python package index or conda with just a few lines of Python code.

All software is available freely on GitHub[5] under permissive open source licenses.

4.1 PAINTERA

Automatic machine learning based neuron reconstruction depends on high-quality training data. Efficient proof-reading of high quality automatic segmentation requires that large datasets can be navigated rapidly and that the consequences of corrections are immediately visible to the user. In order to support the intended use for ground truth generation, I created a comprehensive list of requirements for an appropriate proof-reading tool:

1. Render a set of three orthogonal, two-dimensional cross-sections of arbitrarily large three-dimensional data at arbitrary orientations and zoom lev-

[4] https://github.com/scijava/jgo
[5] https://github.com

els. For efficient loading and continuous user experience, volumetric data should be accessible as small three-dimensional blocks (or chunks) instead of large two-dimensional section series.

2. Support visualization of multi-channel raw data as arbitrary color-composites with configurable contrast range and color-mapped label data that is appropriately overlaid over raw image data.

3. Agglomerate fragments into segments with click-based merge and split actions on fragments.

4. Modify label voxel data with basic brush and flood-fill operations similar to image manipulation software such as MS Paint, GIMP, or Photoshop. Brush strokes should be two-dimensional and aligned with the current cross-section and operate at any zoom level.

5. Create three-dimensional mesh representations of label data as needed on the fly. Pre-computed meshes are not an option as paint and assignment operations change the underlying data. Changes in the segmentation through agglomeration or voxel modifications update the 3D representation immediately.

6. Support multi-scale label data that degenerates gracefully at lower zoom levels so that 3D-visualization still supports meaningful proof-reading interactions.

7. Access to multi-dimensional image processing[6] libraries and computational resources of client computer for performant client-side processing.

8. Easy ways to add extensions like new data types beyond the supported raw and label data, controls, or workflows, *e.g.* active learning proof-reading.

Besides these requirements, familiarity with the main programming language of the tool is a beneficial factor as it leads to cleaner and potentially more efficient code. Ideally, the platform allows for easy distribution on major operating systems (Windows, macOS, Linux distributions).

4.1.1 *Related Work*

Recent years have seen an emergence of a wealth of great visualization and proof-reading tools to support connectome reconstruction for 3D-EM. *CATMAID*[7] (Saalfeld, Cardona, et al. 2009) is a collaborative annotation tool with a focus on tracing neuron skeletons and annotating synaptic connections in arbitrarily large section series. CATMAID focues on tracing and does not support editing dense segmentations.

[6] Volume processing could be considered a more appropriate term.
[7] https://catmaid.org

Ilastik[8] (Sommer et al. 2011) was designed for interactive pixel classification with random-forest classifiers from user annotations. It is extensible and workflows have been added for object classification, tracking of cells or animals, density counting, carving, and multi-cut segmentation. As a Python project, ilastik has access to the many image processing and machine learning libraries that expose a Python interface, *e.g.* *VIGRA*[9] (Köthe et al. 2008), *scikit-image*[10] (Van der Walt et al. 2014), *OpenCV* [11], or *scikit-learn*[12] (Pedregosa et al. 2011). Ilastik visualizes raw and label — albeit scalar — data as orthogonal cross-sections. It favors the HDF5 as data format and supports stacks of various two-dimensional image formats but no format that supports non-scalar label data. Cross-sections cannot be orientated freely and are always aligned with the axes of the data. There is no proof-reading workflow for neuron segmentation and segmentations cannot be visualized in 3D.

Mojo and *Dojo*[13] form a pair of standalone proof-reading client and web-based crowd-sourced annotation tools, respectively (Haehn et al. 2014). The two tools target two different audiences and differ in their respective level of complexity (and thus proof-reading power): Mojo targets expert users and offsers a more complex and powerful toolset; Dojo on the other hand sees strength in numbers: a large number of non-expert users proof-read volumes collaboratively — the user interface thus needs to be simple and intuitive. Mojo was considered for ground truth annotations for the CREMI challenge[14] but painting was not responsive enough for effective annotations. Cross-sections cannot be oriented arbitrarily, and corrections of segmentations have to be made slice by slice. Mojo and Dojo support only two-dimensional fragments and corrections. Note that, at the time of writing of this dissertation, the link to a beta distribution of Mojo 2.0[15] is inactive, and the last commit to the Mojo GitHub repository[16] was made on Nov 18 2013, more than 5 years ago. The build instructions[17] say only "The libraries in Mojo2.0\Mojo\Sdk is not included in the repository - ask Seymour for a zip (download link pending)" — it is thus safe to assume that the development of Mojo has been discontinued.

NeuTu[18] (Zhao et al. 2018) was developed as a tool for proof-reading segmentations in large scale connectomics. It offers split and merge actions to refine segments and 3D fragments. Segments can be visualized in a 3D viewer with polygon meshes (only pre-computed; no support for interactive mesh generation) or as a set of spheres for immediate updates of the 3D visualization when fragments or segments are modified. NeuTu uses seeded watersheds to split fragments; seeds are added as 2D brush strokes or by ray tracing in the 3D view. While

[8] https://www.ilastik.org
[9] https://ukoethe.github.io/vigra
[10] https://scikit-image.org
[11] https://opencv.org
[12] https://scikit-learn.org/stable
[13] https://www.rhoana.org
[14] https://cremi.org
[15] http://people.seas.harvard.edu/~seymourkb/Mojo
[16] https://github.com/Rhoana/Mojo
[17] https://github.com/Rhoana/Mojo/blob/b0d79bc4/BUILD.txt
[18] https://github.com/janelia-flyem/NeuTu

seed points can be painted with a brush, there is no general painting tool for fine-grained changes or annotations without existing segmentations. NeuTu is tightly coupled[19] to the DVID (Katz and Stephen M Plaza 2019) data-service and therefore does not support non-scalar label types. Basic navigation modes — zoom to change the field-of-view, translate within or orthogonal to the cross-section — are supported but cross-sections cannot be oriented arbitrarily. Mipmaps are only supported for generating 3D visualizations of segments more efficiently but not for the two-dimensional cross-sectional view. 2D-rendering is thus not as responsive as it is for viewers that support mipmap-rendering.

BigDataViewer[20] (Pietzsch, Saalfeld, et al. 2015) renders two-dimensional cross-sections of arbitrarily large three-dimensional volumes (and time-series) at arbitrary orientations. At its core, a multi-resolution CPU renderer selects the appropriate level in a mipmap pyramid based on the current zoom level and renders a cross-section through each source converted into ARGB space; these cross-sections are then combined according to customizable composition rules, *e.g.* addition, alpha composition, or others. To further improve performance and guarantee uninterrupted user experience, BigDataViewer supports non-blocking rendering, triple buffering, and CPU rendering at multiple screen scales — lower resolution cross-sections are rendered to update the screen more quickly during navigation. Efficient transformation from local coordinates into viewer space and manipulation of voxels without unnecessary copies of data are made possible by the use of the highly optimizied generic multi-dimensional image processing library ImgLib2[21] (Pietzsch, Preibisch, et al. 2012) for the Java Virtual Machine (JVM),[22] in particular virtualized pixel access, transparent, virtualized image extension, interpolation, coordinate transformations, and caching. BigDataViewer was designed for visualization of selective plane illumination micrographs (Huisken et al. 2004; Pitrone et al. 2013) and consequently does not offer any EM annotation or proof-reading functionality; its extensibility as well as efficient and responsive rendering have inspired visualization and annotation frameworks for connectomics that I will describe in the following.

Neuroglancer[23] uses JavaScript[24] and WebGL[25] for web browser based visualization of volumetric data with a focus on electron micrographs of nervous systems. The GPU accelerated renderer was inspired by the CPU rendering mechanism of BigDataViewer. Neuroglancer meets many of the requirements outlined above: GPU rendering through WebGL makes navigation fast and responsive in multi-scale raw and label data, a 3D viewer renders (pre-computed) meshes of neurons, orthogonal cross-sections can be arbitrarily oriented, and distribution of web-browser based software is simple — users only need a (compatible) web-browser. Neuroglancer is extensible. Certain missing features — merge and split actions on fragments, for example — could be easily implemented; the lack of

[19] NeuTu is referred to as a client of DVID in Zhao et al. (2018).
[20] https://imagej.net/BigDataViewer
[21] https://imagej.net/ImgLib2
[22] Any programming language that runs in the JVM can use ImgLib2.
[23] https://github.com/google/neuroglancer
[24] https://www.javascript.com
[25] https://get.webgl.org

```
> performance.memory.jsHeapSizeLimit
< 2181038080
```

Figure 41: The memory limit of Chrome[26] (version `72.0.3626.119`) on Arch Linux is about 2.2GB per browser tab.

multi-dimensional image (or volume) processing frameworks in JavaScript and limited browser memory (figure 41) and threading support without shared data through web workers impede client-side computations and implementation of certain missing features like multi-scale painting or on-the-fly 3D mesh generation. (Micro-)services would be a viable workaround but would introduce a layer of complexity and send copies of data across processes.

Knossos[27] is a Qt5[28] based tool for visualization of volumetric data with support for skeleton tracing, dense annotations, and proof-reading. Three views render orthogonal 2D slices that are axis-aligned with the data at arbitrary zoom levels. An optional 2D slice can be rendered at arbitrary orientations in an additional view. Skeletons and the orthogonal slices are visualized in a 3D view. Dense annotations are modified with 2D brush strokes that are axis-aligned with the data. Precomputed meshes can be loaded from Polygon File Format files for visualization. Mesh generation on-the-fly is supported on the current development branch but restricted to data blocks that have already been loaded into memory. Knossos was initially used to generate the dense CREMI neuron annotations but did not produce satisfactory results. *webKnossos*[29] (Kevin M Boergens et al. 2017) is a similar, web-browser based tool for tracing and proof-reading. In addition to the regular Knossos features, it introduces a novel "flight" mode for more focused tracing and supports visualization and on-the-fly generation of 3D iso-surface meshes as an experimental feature.

BigCAT[30] is a tool for the fast and efficient annotation of dense neuron anatomy and synaptic clefts and partners. We created it after previous attempts to generate the dense CREMI neuron annotations with a Knossos based workflow had saturated at an unsatisfying level of correctness. BigCAT uses BigDataViewer to render label data on top of raw data. It can be considered the immediate ancestor to Paintera and I initially developed many of the features of modern Paintera in BigCAT, *e.g.* brush strokes (3D sphere) and flood-fill painting tools. BigCAT was a targeted solution for the annotation of relatively small data sets (albeit still the largest amount of publicly available ground truth for 3D-EM connectomics to date) and as such has a narrow focus on the supported format (HDF5) and is not designed for extensibility or large datasets.

The features and shortcomings of these tools have inspired the architecture of Paintera, which I will describe in the following.

[27] https://knossos.app
[28] https://www.qt.io
[29] https://webknossos.org
[30] https://github.com/saalfeldlab/bigcat

4.1.2 *Architecture & Design*

In recent years, I have been using languages that run in the JVM in professional settings almost exclusively — Java (and JVM based languages) is now the programming language that I used professionally more than any other language — and I have become an expert in the Java multi-dimensional image processing framework ImgLib2. Considering my professional focus on the JVM and the review of related software (section 4.1.1), it was an obvious decision to build Paintera around the BigDataViewer multi-resolution renderer (Pietzsch, Saalfeld, et al. 2015, Supplementary Note 2) within the Java environment. A beneficial side-effect is ease of distribution — the JVM adds a layer of abstraction and is its own platform that is agnostic of the operating system it runs on: code compiled on one computer will run on any other computer, which is generally not true for native languages.

The BigDataViewer graphical user interface (GUI) is built on top of the Java GUI toolkit Swing. I chose to develop Paintera in the most recent Java GUI framework, JavaFX, because it offers a more modern design, *e.g.* a scene graph, event capturing and bubbling phases, and built-in support for 3D rendering.

After experiments had established that including Swing elements like the core GUI element of BigDataViewer — the `ViewerPanel` — in a JavaFX GUI is not very performant, I decided to re-implement the multi-resolution renderer for JavaFX based GUI elements that would be accessible for rendering through a JavaFX port of the `ViewerPanel` — the `ViewerPanelFX`. The major difference between single and multiple cross-sections is how data sources are transformed affinely from local voxel space into the viewer plane. Pietzsch, Saalfeld, et al. (2015, Supplementary Note 2) define two transforms for that purpose:

- T_i maps the local voxel coordinates of the i^{th} source into a global coordinate space — an arbitrarily defined isotropic $3D$ coordinate system. This can be used to, for example, consider the resolution of a source or a relative offset between sources.

- V maps from global coordinate space to the viewer space.[31] This transform also accounts for changes of the size of the viewer window — when resizing the viewer window, the field-of-view should not change.

Each source is then mapped into viewer space by concatenation of these transforms VT_i. It is clear that, in order to support multiple synchronized cross-sectional views, V would need to be decomposed into a shared component and components that are specific to each individual view:

- G is the shared or *global* component. Changes to the orientation, position, or field-of-view of the set of cross-sections should modify this transform accordingly.

[31] Note that Pietzsch, Saalfeld, et al. (2015) use T^{VW} for this transform. I changed the name to avoid ambiguity with the local-to-global transform of each source T_i, in particular when using multiple cross-sectional views.

Grid Cell	Contents
top left	XY cross-sectional view
top right	ZY cross-sectional view
bottom left	XZ cross-sectional view
bottom right	3D scene

Table 10: Alignment of cross-sectional views and 3D scene in a two-by-two grid.

- O_i defines the orientation of a cross-section and should be considered constant. For orthogonal cross-sections, there are three possible configurations (the translation components of the affine transformations are zero and therefore omitted):

$$O_{XY} = \begin{bmatrix} +1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & +1 \end{bmatrix} \tag{77}$$

$$O_{ZY} = \begin{bmatrix} 0 & 0 & +1 \\ 0 & +1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{78}$$

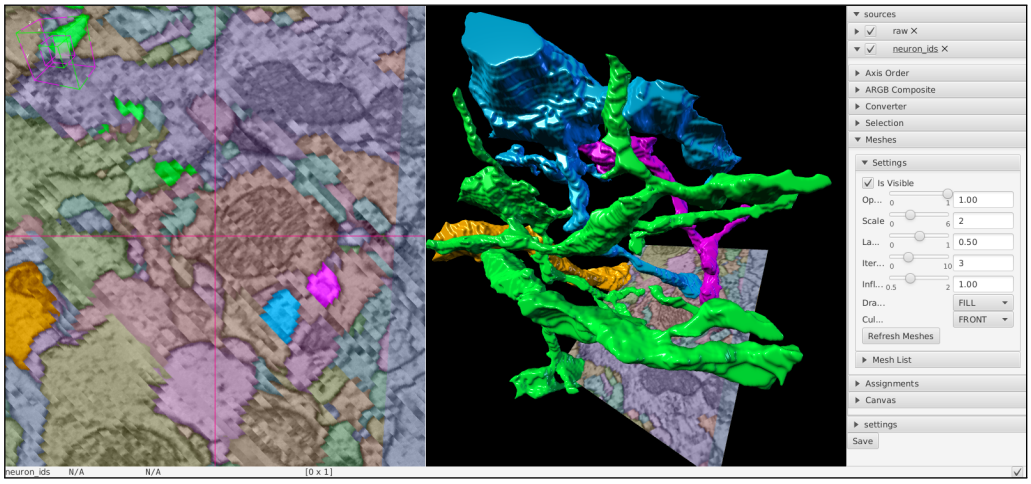
$$O_{XZ} = \begin{bmatrix} +1 & 0 & 0 \\ 0 & 0 & +1 \\ 0 & -1 & 0 \end{bmatrix} \tag{79}$$

  The indices XY, ZY, and XZ indicate the orientation of each cross-section in world space if the global transform G is diagonal, i.e. it does not have any rotation component.

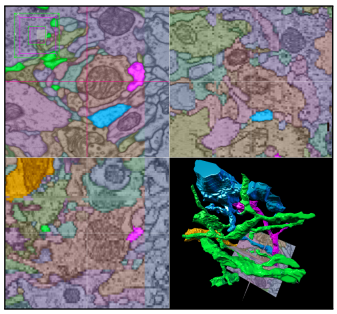
- S_i accounts for changes of the size of the viewer window.

The transformation V_i of each individual viewer $i \in \{XY, ZY, XZ\}$ is the concatenation $V_i = S_i O_i G$. All V_i need to be updated whenever G changes. User inputs to change the orientation or field-of-view of the cross-sections changes only G, while S_i and O_i remain unaffected. Overlays over the cross-sections are rendered independently but are aware of the viewer transform V_i: Cross-hairs indicate the intersection point of the cross-sections and higlight the currently active viewer — if any — through color-coding, wire-frame representations visualize orientation of sources relative to the cross-section, and a circle represents the brush size when painting. The viewers are arranged in a two-by-two grid (`GridPane`[32]) such that neighboring views share a coordinate axis (table 10). The remaining grid cell holds a 3D scene for visualization of currently selected segments (figure 42). To help orientation, all visible 2D cross-sections can be embedded into the 3D scene. The individual cells are resizable and modes to display only the XY cross-sectional view can be toggled (table 11): alone (figure 42(d)) or alongside the 3D scene (figure 42(a)). A status bar at the bottom bar holds information such as the current location of the mouse cursor (in viewer space and global space) and the value under the mouse cursor of the currently selected source. A preferences

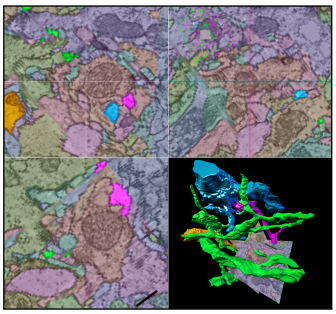
[32] `javafx.scene.layout.GridPane`



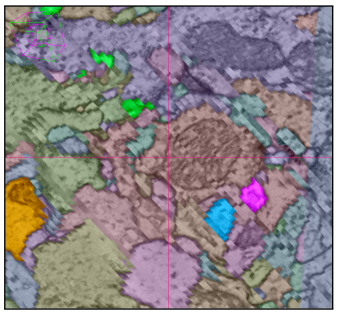
(a) Paintera shows a maximized 2D cross-section at arbitrary orientation and the 3D view with the status bar at the bottom and the preference pane on the right.



(b) All 2D cross-sections (axis aligned) and the 3D view are visible.



(c) All 2D cross-sections at arbitrary orientation and the 3D view are visible.



(d) Maximized view of the single *XY* cross-section at arbitrary orientation.

Figure 42: Layout of the cross-sectional views and 3D scene. Each image captures the entire Paintera window without mouse pointer or operating system specific window titlebar and borders. The sources are the datasets `volumes/raw` and `volumes/labels/neuron_ids` of the padded sample B of the CREMI challenge.[33] Currently selected segments are highlighted in cross-sections and rendered in the 3D scene. Meshes are generated on the fly and memory-cached.

pane on the right contains a list of all sources with source-specific settings and configuration of Paintera (see section 4.1.2.1). The visibility of both the status bar and preferences pane can be toggled (figure 42). Sources can be added as command-line parameters or via the opener dialog. The single, currently active source can have additional event handlers for user input on top of the navigation and controls that are always active (tables 11 to 13).

4.1.2.1 *Preference Pane*

The preference pane consists of two collapsible main sections: The *sources* section lists all sources with individual settings per source. The settings depend on the type of source, but all sources expose common configurations. The *settings* section exposes general settings for Paintera. In addition, a save button stores the current

state of a Paintera project in the project directory. The prefernce pane is toggled by pressing the P key and can be resized by dragging its left border.

Expanding the sources section reveals a list of collapsible panes for each source. A check-box to the left of the source name toggles visibility; the cross-mark to the right removes the source from Paintera after confirmation by the user. Sources are composed in the order — top to bottom — in which they are listed. The order can be re-arranged by dragging and dropping individual sources. This is particularly useful if sources were loaded in the wrong order, *e.g.* raw data was loaded after label data. While each source type provides a set of its own specialized settings, the following settings are generally available:

AXIS ORDER re-maps the spatial axes of the source onto the spatial axes of the arbitrary global coordinate system. By default, trivial correspondence is assumed, i.e. the transformation from local source coordinates into the global coordinate system has only scaling and translational components.

ARGB COMPOSITE changes the composition mode (section 4.1.2.2) for this source.

CONVERTER exposes settings for conversion from the source data type into ARGB color space such as contrast, alpha value, or false color. These are specific for each source type (section 4.1.5).

Some source types may offer additional configuration beyond these standard settings, *e.g.* 3D polygon mesh settings for label sources. The general settings change the look-and-feel, help with navigation, or improve the rendering performance:

NAVIGATION Displays and sets the intersection point of cross-sections as indicated by the crosshairs. The check box to the right of "Rotations" toggles rotation navigations. Rotation speeds for key navigation can be set in "Key Rotation Speeds".

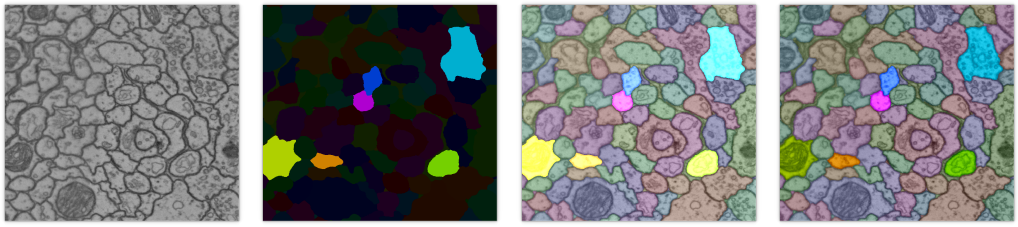
CROSSHAIR The check box toggles crosshair visibility. Color highlights for on/off focus views can be adjusted through dialogs.

ORTHO-VIEWS Toggle visibility of individual or all cross-sections in 3D scene.

3D VIEWER Toggle the 3D scene. If 3D representations are irrelevant, this can boost overall performance.

SCREEN SCALES In order to update the screen as rapidly as possible, cross-sections are rendered at multiple scales. The screen scales are a strictly monotonously decreasing sequence s_n of numbers between zero (exclusive) and one (inclusive): $s_n \in (0, 1] \wedge s_{n+1} < s_n$. These scales can be set to accomodate to different hardware: The number of pixels to be CPU rendered and composed is proportional to the square of a screen scale. For HiDPI monitors, for example, I typically set the maximum screen scale to 0.5; this means that only a quarter of the pixels needs to be rendered compared to a screen scale of 1.

MEMORY Set the cache size that is used for data loading.



(a) Raw data only (b) Copy (c) Alpha add (d) Alpha YCbCr

Figure 43: Composition modes of label data over raw data (a): Copy overwrites the raw data (b), alpha addition oversaturates easily (c), alpha addition in YCbCr space yields the best contrast for label data over raw data (d). Selected ids are highlighted with a higher alpha value.

4.1.2.2 *Composition Modes*

As laid out by Pietzsch, Saalfeld, et al. (2015, Supplementary Note 2), sources can have arbitrary data types that need to be rendered into ARGB space before composition. The converters are source-specific and are explained in detail in section 4.1.5. Visible sources — as indicated by the check box next to their names in the preferences pane — are rendered and composed in the order listed in the preference pane. Paintera offers the following composition types that can be selected in the *ARGB Composite* section of the settings for each individual source (note that compositions are defined as pixel-wise operations and alpha values may vary across pixels depending on the converter):

COPY overwrites the ARGB value disregarding the current value, if any. This is only meaningful for the first source. Raw data is typically added as the first source and defaults to this composition mode.

ALPHA YCBCR combines the Y-channel of current value with the Cb and Cr channels of the ARGB-mapped source, and mixes the result into the current value weighted by the current source's alpha value. This is the default composition mode for label data.

ALPHA ADD multiplies the ARGB-mapped source by its alpha value and adds it to the current value. This is useful for composing multi-color raw data, *e.g.* boundary or affinity predictions, onto raw data.

The composition modes are visualized in figure 43 for raw and label sources.

4.1.3 *Paintera Project*

Paintera can persist its state into a Paintera project (an N5 container) for re-use in subsequent sessions. The Paintera state is serialized into a Json object that is stored in the key `"paintera"` in the attributes of the root group of the Paintera project. The structure of the serialized state (figure 44) resembles the layout of the preference pane (section 4.1.2.1). The attributes file is human-readable and can be modified, *e.g.* to add sources that cannot be added yet through the Paintera GUI. The location of the container can be specified at start-up time, either via command

line argument, or via GUI dialog if no argument is specified.[34] (section 4.1.4) Each source knows the location of the project container and can use it to store temporary data, *e.g.* canvases for painting (section 4.1.5.4). Additionally, a lock file located at `.paintera/lock' relative to the Paintera project container is used to prevent multiple Paintera instances from accessing the same project simultaneously.

4.1.4 *Installation & Synopsis*

Paintera depends on JDK 8[35] and Apache Maven.[36] The open source alternative OpenJDK[37], requires manual installation of JavaFX. OpenJDK, JavaFX, and Maven can be installed through the package managers of many Linux distributions, for example:

```
# Arch Linux
pacman -S jdk8-openjdk
pacman -S java-openfx
pacman -S maven
# Ubuntu
apt install default-jre default-jdk
apt install maven
apt install openjfx
# JavaFX installation on Ubuntu 18.04
apt install \
    openjfx=8u161-b12-1ubuntu2 \
    libopenjfx-java=8u161-b12-1ubuntu2 \
    libopenjfx-jni=8u161-b12-1ubuntu2
apt-mark hold openjfx libopenjfx-java libopenjfx-jni
```

On Ubuntu 18.10 and newer, the bionic repositories are required for the installation of JavaFX.[38]

At the time of writing this dissertation, the ImgLib2-cache dependency used for disk-caching is incompatible with more recent versions of Java than Java 8 due to internal changes to garbage collection. Once all upstream dependencies are compatible with more recent Java versions, Paintera will be updated to use a more recent Java version. In particular, JavaFX will be available as a maven dependency for more recent Java versions and will not be required to be installed in advance. The easiest way to install Paintera is through *conda*:[39]

```
conda install -c conda-forge -c hanslovsky paintera
```

It is important to note that OpenJDK available through the conda-forge channel on conda does not provide JavaFX and thus should not be installed into the same conda environment as paintera. The `paintera` conda package provides a `paintera` executable with the following synopsis:

```
paintera [jgo/jvm arguments --] [Paintera options] [PROJECT]
```

[34] Currently, the project directory of a running Paintera instance cannot be changed but there are plans to make this possible in the future.

[35] https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

[36] https://maven.apache.org/

[37] https://openjdk.java.net/install

[38] https://bugs.launchpad.net/ubuntu/+source/openjfx/+bug/1799946

[39] https://conda.io

```
{
  "n5": "2.0.2",
  "paintera": {
    "sourceInfo": {
      "sources": [
        {
          "type": "fully.quallified.class.Name",
          "state": {}
        }
      ],
      "currentSourceIndex": 0,
      "numSources": 1
    },
    "globalTransform": [
    1.0, 0.0, 0.0, 0.0,
    0.0, 1.0, 0.0, 0.0,
    0.0, 0.0, 1.0, 0.0
    ],
    "windowProperties": {"width": 1920, "height": 2062},
    "gridConstraints": {
      "previousFirstRowHeight": 50.0,
      "previousFirstColumnWidth": 50.0,
      "isFullScreen": false,
      "firstRowHeight": 50.0,
      "firstColumnWidth": 50.0
    },
    "crosshairConfig": {
      "onFocusColor": "#FF0088FF",
      "offFocusColor": "#FFFFFFFF",
      "isVisible": false
    },
    "orthoSliceConfig": {
      "enabled": true,
      "showTopLeft": true,
      "showTopRight": true,
      "showBottomLeft": true,
      "delayInNanoSeconds": 200
    },
    "navigationConfig": {
      "allowRotations": true,
      "buttonRotationSpeeds": {
        "slow": 0.5,
        "regular": 5.0,
        "fast": 45.0
      }
    },
    "viewer3DConfig": {"areMeshesEnabled": true},
    "screenScalesConfig": {"scales": [1.0, 0.5, 0.25]}
  }
}
```

Figure 44: Example Paintera project with a single source. Note that the source is not a valid source but a Json schema: the `"state"` Json object is deserialized as an instance of class `"type"`. See section 4.1.5 for real examples of serialized states.

```
git clone https://github.com/saalfeldlab/paintera.git
cd paintera
mvn clean install
jgo \
    [jgo arguments] \
    -r imagej.public=https://maven.imagej.net/content/groups/public \
    org.janelia.saalfeldlab:paintera:<VERSION>+org.slf4j:slf4j-simple:1.7.25
```

Figure 45: Install and run the the latest development version of Paintera. The example expects `git` to be installed on the system. If that is not possible, the source code can be downloaded as zip archive from GitHub. See section 4.4 for more details (including installation) for jgo. Replace `<VERSION>` with the version you installed, *e.g.* `0.11.1-SNAPSHOT`.

For a list of available Paintera options, invoke paintera with the `--help` flag. See section 4.4.1 for available `jgo` arguments and `java -help` for `jvm` arguments. By default, Paintera uses half the system's memory as maximum Java heap size. The `-Xmx` JVM option overrides that value. The latest development version is available on the `master` branch of the official Paintera GitHub repository[40]. Building and running the current development requires only a few simple commands (figure 45).

4.1.5 *Supported Data*

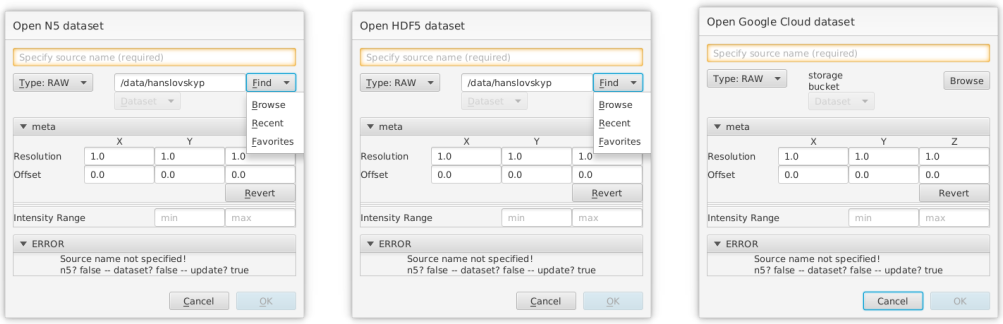
While Paintera is extensible with custom data types, the current focus is proofreading 3D-EM connectomics and thus, by default, single- and multi-channel raw and label data are supported through the N5-API. Data can be added via the UI (section 4.1.5.1) or by modifiying the Paintera project. Additional specialized source types are available for the visualization of synaptic clefts from synaptic cleft predictions (Heinrich, Funke, et al. 2018). Currently, these source types can not be added in the UI. Instead, they have to be added by manipulation of the Paintera project. While performance is optimal when using Paintera-specific datasets stored as filesystem-based multi-resolution N5, HDF5 and Google Cloud are supported as well. By convention, resolution and an offset in an arbitrary global coordinate system of a data set is specified with the `"resolution"` and `"offset"` attributes.

Unsupported data types or data backends can be made available through Paintera extensions (section 4.1.7).

4.1.5.1 *Open Dataset Context Menu*

Datasets can be added through a context menu at the current location of the mouse cursor (table 11). The context menu lists the supported data backends and can be extended with custom entries to load arbitrary kinds of data (section 4.1.7). For the supported-by-default "N5", "HDF5", and "Google Cloud" menu entries, a dialog prompts for the selection of container and dataset as well as meta settings (figure 46). After confirmation of the selection with the "OK" button, a data source is appended to Paintera. The newly added source is rendered on top of the

[40] https://github.com/saalfeldlab/paintera



(a) N5 Filesystem (b) N5 HDF5 (c) N5 Google Cloud

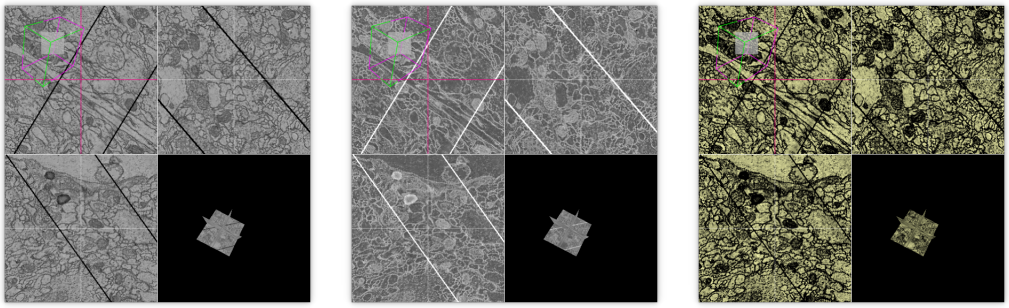
Figure 46: Dataset dialogs for supported N5 backends. Thanks to the shared N5-API, the dialogs differ mainly in the container prompt: N5 filesystem (a) and N5 HDF5 (b) dialogs support mnemonics that are triggered with the `ALT` key and find containers via a file browser or from recently used containers or favorite locations defined in the paintera config at `~/.config/paintera.yml`, the Google Cloud backend prompts for required meta data and authentication when clicking the "Browse" button (c). Once a valid container is selected, available data sets are discovered automatically — Paintera data sets, regular datasets and legacy multi-scale groups are considered. Once discovery has finished, datasets can be selected via the "Dataset" drop-down menu with fuzzy-matching. The name field is then auto-populated as the last segment of the dataset name. The "Type" drop-down specifies the data type (raw or label). Resolution and offset are read from the dataset, if available, and can be modified as needed. For raw data, the contrast can be specified with the intensity range that is initialized from the dataset attributes or with meaningful defaults.

already existing sources using a reasonable default composition mode (figure 43).

4.1.5.2 *Paintera Datasets*

In order to use Paintera to its full potential, the conversion of single-scale datasets into the preferred Paintera dataset format is recommended. At its core, the Paintera dataset is a group in an N5 file system container with a "painteraData" attribute containing a Json object that specifies the data type as string at the "type" key. Currently, "raw" (section 4.1.5.3), "label" (section 4.1.5.3), and "channel" are possible options.[41] All paintera dataset groups must contain a multi-scale subgroup "data" with boolean attribute "multiScale" set to "true" and optional floating-point array attributes "resolution" and "offset" that specify how to map local coordinates into a global coordinate space as well as data type specific attributes. The multi-scale "data" group contains datasets "`s0`" to "`sN`" for the levels of the mipmap pyramid, where "`s0`" is the highest resolution data set. Each dataset must have the floating point array attribute "downsamplingFactors" relative to the "resolution" attribute of the "data" group: If not specified, "downsamplingFactors" will default to identity. Besides the "data" group, any ad-

[41] At the moment, the open-dataset dialog for N5 file system containers (section 4.1.5.1) only tests for presence of the "painteraData" attribute to determine if a group is a Paintera dataset but ignores the "type" for pre-selecting the data type.



(a) $t_R = t_G = t_B = 255$, $m = 0$, $M = 255$ (b) $t_R = t_G = t_B = 255$, $m = 255$, $M = 0$ (c) $t_R = t_G = 255$, $t_B = 179$, $m = 100$, $M = 180$

Figure 47: Raw data with various converter settings: default color and contrast range for 8-bit unsigned integer data (a), inverted (b), and with different color and increased contrast (c). Full opacity ($c_A = 255$) is used for all settings.

ditional subgroup, dataset or plain directory can be added to the Paintera dataset group as needed by specific data types. This is used, for example, to maintain an index for efficient lookup of all blocks (chunks) that contain a label for each label (section 4.1.5.4). While Paintera datasets are preferred, loading data as plain N5 datasets or multi-scale groups is supported as well. Custom data types that are added as extensions may or may not follow this convention. It is the extension's responsibility to make sure that data is formatted and added appropriately.

4.1.5.3 *Raw Data*

Raw data generally refers to three-dimensional micrograph volumes but, practictally, any scalar-valued, non-categorical data like boundary predictions or image features can be considered raw data and is treated as such in Paintera. Raw data can be added through the open dataset context menu (section 4.1.5.1) by selecting the "RAW" type (figure 46). Voxel values v are mapped into 32-bit ARGB color c by scaling each color channel of a target color t by the voxel value relative to a contrast range $[m, M]$:

$$c_i = \min\left\{255, \max\left\{0, \left\lfloor t_i \times \frac{v - m}{M - m} \right\rceil\right\}\right\} \forall i \in \{R, G, B\}, \tag{80}$$

where $\lfloor x \rceil$ denotes rounding to the nearest integer. Setting $M < m$ inverts the color mapping, *i.e.* black areas appear as white and vice versa. Figure 47 shows the effect for different choices of target color t and contrast range $[m, M]$. The opacity or alpha component c_A is a separate parameter that is used for alpha color composition (section 4.1.2.2) of raw data, *e.g.* for overlaying boundary predictions. The conversion parameters can be adjusted in the "Converter" pane (figure 48) of the settings for a raw source inside the preference pane (section 4.1.2.1). Pressing the I-key toggles between tri-linear and nearest-neighbor interpolation.

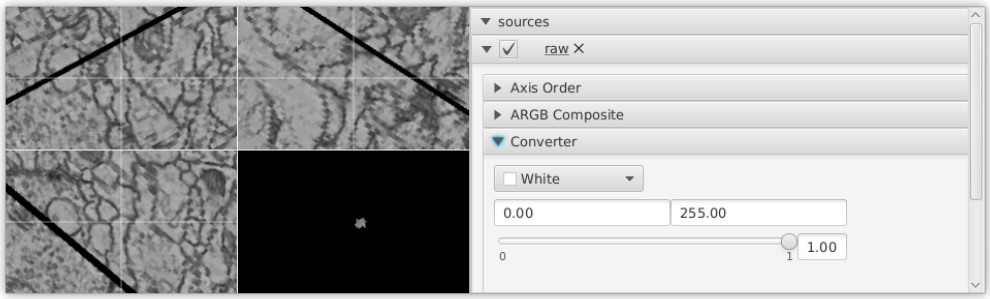
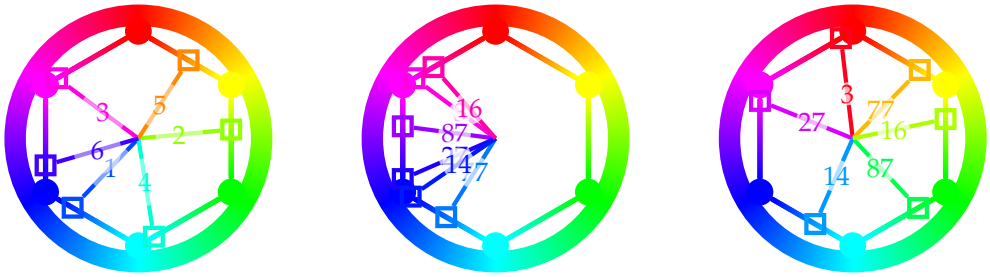


Figure 48: Converter pane for raw data with color dialog for t, text fields to set contrast range $[m, M]$, and a slider for opacity c_A. The selected opacity $o \in [0, 1] \subset \mathbb{R}$ is mapped linearly into $[0, 255] \subset \mathbb{Z}$ by multipliaction and rounding: $\lfloor 255 \times o \rceil$

4.1.5.4 *Label Data*

In image instance segmentation in general and neuron reconstruction in particular, individual voxels are grouped into objects with *label* ids, unique integer-valued identifiers. Typically, label ids are positive integers $\mathbb{Z}^+$ with an additional zero-valued label indicating background. 64-bit unsigned integer types are preferred over smaller integer representations to guarantee a sufficient number of unique label ids: 32-bit unsigned integer types can represent $2^{32} - 1 \approx 4.3 \times 10^9$ non-background label ids; for 64-bit unsigned integer, that number is 10 orders of magnitude larger with $2^{64} - 1 \approx 1.8 \times 10^{19}$.

For visiual distinction of individual fragments, label ids are mapped into ARGB color space with a lookup table that cannot be pre-calculated because memory requirements may exceed system limitations or the number of label ids may be unknown a-priori or variable. Consequently, it is impossible to guarantee uniform distribution of the label ids across RGB space. Calculating pseudrandom for a virtual pseudorandom number stream comes at a steep computational cost depending on the label id. Instead, Paintera distributes label ids around the fully-saturated color circle in steps of the golden ratio $\Phi = 2(1 + \sqrt{5})^{-1}$ (figure 49). Different sets of label ids can have different distributions onto the color wheel, *e.g.* clustered within a small subset of the color wheel (figure 49(b)), and an integral scaling factor s is applied to the label id to allow for modification of the color scheme if objects are not distinct enough in ARGB color space. The scaling factor can be incremented by one, decremented by one, or set specifically (table 13). For efficiency, instead of picking a hue value in HSV space and subsequent conversion into RGB space, colors are interpolated from six equidistant colors on the color wheel (HTML color codes in parantheses): red (`#FF0000`), yellow (`#FFFF00`), green (`#00ff00`), cyan (`#00ffff`), blue (`#0000ff`), and magenta (`#ff00ff`). the



(a) $s = 1$ (b) $s = 1$ (c) $s = 7$

Figure 49: Parameterized golden angle color assignment for label ids: The six base colors are enscribed into a color wheel. A choice of $s = 1$ for the seed parameter nicely separates label ids $\{1, 2, 3, 4, 5, 6\}$ on the color wheel(a) but label ids $\{3, 14, 16, 27, 77, 87\}$ are clustered within only a small subset of the color circle(b). A different choice for the seed parameter $s = 7$ achieves better separation for the latter set of label ids(c).

color $c = (r, g, b)$ that is assigned to a label j is then an interpolation of consecutive rows of the color matrix c:

$$c = \begin{bmatrix} 255 & 0 & 0 \\ 255 & 255 & 0 \\ 0 & 255 & 0 \\ 0 & 255 & 255 \\ 0 & 0 & 255 \\ 255 & 0 & 255 \\ 255 & 0 & 0 \end{bmatrix} \tag{81}$$

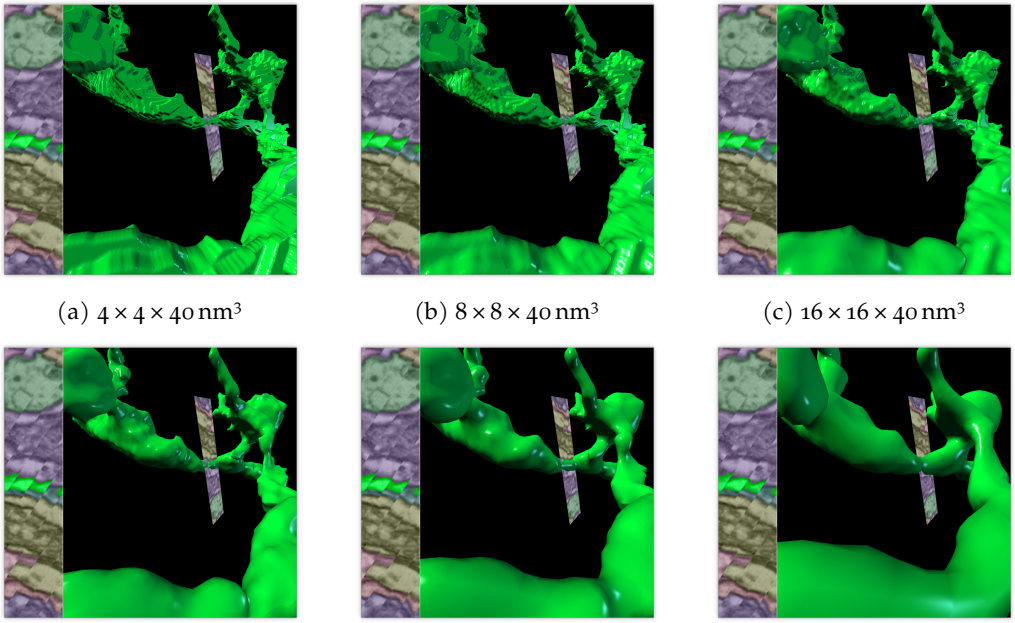
$$p = 6 \times (s \times j \times \phi - \lfloor s \times j \times \phi \rfloor) \tag{82}$$

$$w = \lceil p \rceil - p \tag{83}$$

$$c_i = \lfloor w c_{\lceil p \rceil, i} + (1 - w) c_{\lceil p \rceil + 1, i} \rceil \forall i \in \{1, 2, 3\}, \tag{84}$$

where $\lceil x \rceil$ denotes the ceiling function that maps a real number x to the smallest integer greater than or equal to x. The last row of the color matrix is a duplicate of the first row to avoid range checks before interpolation. A fragment can be colored either with its own label id or with the id of the segment it is assigned to. Individual fragments can be de-/selected via left click onto the 2D cross-sectional views. Right click toggles selection of a fragment without deselecting other selected fragments. Selected fragments are highlighted with a higher opacity and all fragments that are in the same segment are highlighted with an opacity between explicitly selected fragments and regular opacity. Regular, selected fragment, and selected segment opacities are exposed as sliders in the source-specific settings inside the preference pane. The segment id of the last selected fragment can be locked/unlocked with the `l` to indicate (temporary) completion for specific neurons. All fragments of a locked segment are rendered with zero opacity to help proof-readers focus on unfinished segments.

As outlined in section 3.1, automatic neuron reconstruction usually is a multi-step process: machine learning algorithms predict boundaries within the data, voxels are grouped into fragments, and, finally, fragments are assembled into segments. The proof-readinger experience thus benefits from tools that are aware of



(a) $4 \times 4 \times 40\,\text{nm}^3$ (b) $8 \times 8 \times 40\,\text{nm}^3$ (c) $16 \times 16 \times 40\,\text{nm}^3$

(d) $32 \times 32 \times 40\,\text{nm}^3$ (e) $64 \times 64 \times 80\,\text{nm}^3$ (f) $128 \times 128 \times 160\,\text{nm}^3$

Figure 50: Triangle meshes for neuron 568025 of CREMI sample B at multiple mipmap levels $0 \leq l \leq 5$, ordered from highest (a) to lowest (f) resolution, with voxel resolutions starting at a nominal resolution of $4 \times 4 \times 40\,\text{nm}^3$. The anisotropy of the data is apparent in the highest resolution mipmap level (a) and the mesh appears to be a composed of discs. After multiple levels of anisotropic down-sampling, a near-isotropic resolution is reached in (d). The right choice for the mesh representation is usually a compromoise between computational efficiency and the desired level of detail.

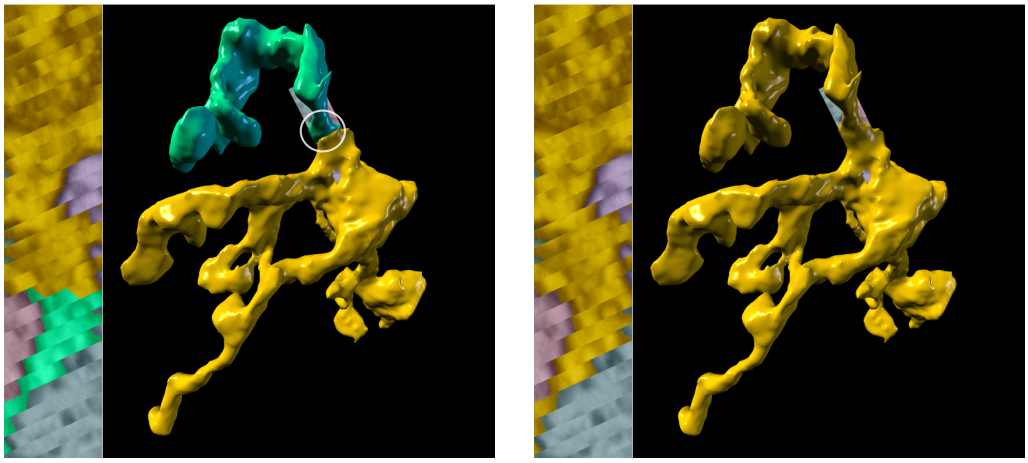
and exploit this structure. In the following, I will explain how Paintera can be utilized to create or modify existing fragment-segment agglomerations represented as assignments or lookup tables and to correct fragments or create segmentations from scratch with various painting operations.

3D-REPRESENTATION Segments are represented in the 3D viewer as surface-rendered triangle meshes. In order to allow for interactive updates of meshes when segments are modified by merge/detach actions or painting, meshes cannot be precomputed. Instead, meshes are generated on request with the marching cubes algorithm (Lorensen and Cline 1987). Several measures are implemented to ensure fast and responsive generation of triangle meshes:

1. Only relevant blocks that contain at least one voxel of the requested segment are processed. To that end, Paintera maintains an index for efficient lookup of all blocks (chunks) that contain a label for each label.

2. Meshes can be generated at any scale of the mipmap pyramid. This is in particular useful for larger datasets (less data to process and smaller mesh size) and for anisotropic data (generate meshes at isotropic resolution).

3. A memory cache avoids unnecessary repeated generation of meshes.



(a) 3D representation of two fragments with notable boundary.

(b) 3D representation of segment after merging two fragments.

Figure 51: Automatic update of meshes after merge: Two neighboring fragments are selected and displayed in 3D (a). The boundary surface between the two fragments is notable as indicated by the white ellipsoid overlay. After merging the two fragments into a single segment, the mesh is updated automatically (b). The formerly visible boundary between the fragments in the 3D representation has vanished as the mesh is generated for the complete segment rathern than a combination of individual fragment meshes.

The index of blocks that contain a label (1) is provided in the "label-to-block-mapping" directory of a Paintera label dataset. While mesh support is available for other label datasets, mesh generation without this index can be slow because the entire dataset has to be processed and multi-scale data may not be available. Conversion into Paintera datasets is therefore highly recommended (section 4.1.5.5). Mesh settings like the mipmap level ("Scale") or smoothing settings ("Lambda", "Iterations") and a list of current meshes are exposed in the "Meshes" pane of a label source within the preference pane (section 4.1.2.1). Meshes use the global mesh settings by default but can be set to use specific settings. Whenever the fragment-segment assignment is modified through merge or detach actions, meshes of affected segments are updated automatically. After painting, in contrast, the mesh update is triggered manually by the user (R or the "Refresh Meshes" button in the mesh settings) instead.

FRAGMENT-SEGMENT AGGLOMERATION Agglomeration or assignment of fragments into segments is a function A that maps from fragment label id space $F \subseteq \mathbb{Z}^+$ into segment label id space $S \subseteq \mathbb{Z}^+$:

$$\begin{aligned} &\mathrm{A}: F \mapsto S \\ &\forall s \in S : s = \mathrm{A}(s) \iff \nexists \hat{s} \in S : \mathrm{A}(\hat{s}) = \mathrm{A}(s) \end{aligned} \tag{85}$$

In particular, this means that, when two single-fragment segments are merged, a new unique label id $l^* \in \mathbb{Z}^+ \setminus (F \cup S)$ must be created. In practice, it is easy to track the maximum label. A new unique label is the smallest positive integer that is larger than the maximum label:

$$l^* = 1 + \max(F \cup S) \tag{86}$$

While there might be unused labels

$$U = \{u \in \mathbb{Z}^+ \setminus (F \cup S) : \min\{F \cup S\} < u < \max\{F \cup S\}\}, \tag{87}$$

and it might appear wasteful to disregard those, it is unlikely that all possible will be used up in a realistic scenario: Exhausting all available label ids provided by an 64-bit unsigned integer type would take more than 500 years at an unrealistically fast request rate of one unique label id per nanosecond:

$$t = \frac{2^{64}}{1/\text{ns}} = \frac{2^{64}}{10^9}\text{s} \approx 585\text{yr} \tag{88}$$

Multiple fragments can be assigned to the same segment, and A is thus not invertible. It is still useful to define $\bar{\text{A}}^{-1}$ mapping from S to the subspace $F_s \subseteq F$ of all fragments contained in a segment:

$$\begin{aligned} &\bar{\text{A}}^{-1} : S \mapsto \{F_s \subseteq F : s \in S\} \\ &\bigcup_{s \in S} F_s = F \\ &F_j \cap F_k = \emptyset \forall j \neq k \end{aligned} \tag{89}$$

The fragment-segment mapping A is implemented as a lookup table in Paintera. It is initialized from the "fragment-segment-assignment" dataset inside the root group of the source if the data is provided in Paintera format, or identity $\text{A}(f) = f \forall f \in F$ otherwise. The following assignment actions are applied on top of the initial lookup to create an updated lookup table A*:

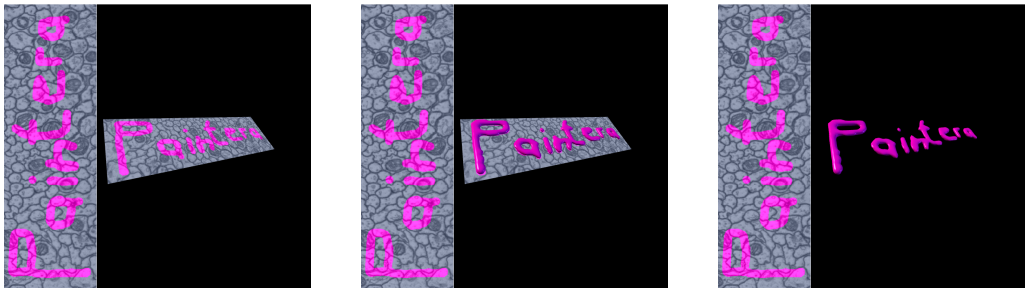
MERGE actions combine two segments as the union of all contained fragments. The segments are identified by the currently selected fragment — the last selection in case of multiple selections — and a second fragment specified with `shift` left click:

$$\begin{aligned} &f_1, f_2 \in F : \text{A}(f_1) \neq \text{A}(f_2) \\ &s_i = \text{A}(f_i) \\ &\text{A}^*(f \in F_{s_1}) = \text{A}^*(f \in F_{s_2}) = \begin{cases} \text{A}(f_1) & \text{if } \text{A}(f_1) \neq f_1 \\ \text{A}(f_2) & \text{if } \text{A}(f_2) \neq f_2 \\ 1 + \max\{F \cup S\} & \text{otherwise.} \end{cases} \end{aligned} \tag{90}$$

DETACH or *split* actions remove individual fragments from a segment. `shift` right click identifies the fragment $f_r \in F$ to be removed from the segment containing the currently selected fragment $f_s \in F : \text{A}(f_s) = \text{A}(f_r) \wedge f_s \neq f_r$:

$$\begin{aligned} &\text{A}^*(f_r) = f_r \\ &\text{A}^*(f_s) = \begin{cases} f_s & \text{if } |F_{\text{A}(f_s)}| = 2 \\ \text{A}(f_2) & \text{otherwise.} \end{cases} \end{aligned} \tag{91}$$

Note that the detach action is not the inverse of a merge action: If two segments $j \neq k : |F_j| > 1$ are merged into $F_j \cup F_k$ by representative fragments $f_j \in F_j$ and



(a) (b) (c)

Figure 52: The word "Paintera" is painted into a cross-section of an empty label dataset over raw data (a). The meshes are available after explicitly re-freshing the mesh cache (b). Disabling the ortho-slices in the 3D view demonstrates that an actual 3D representation of the painted labels is created and rendered (c).

$f_k \in F_k$ and, subsequently, fragment f_j is detached from $F_j \cup F_k$, then the resulting segments $\{f_j\}$ and $F_j \cup F_k \setminus f_j$ are not equal to F_j and F_k:

$$\{\{f_j\}, F_j \cup F_k \setminus f_j\} \neq \{F_j, F_k\} \tag{92}$$

This becomes obvious as

$$f_j \in F_j, \tag{93}$$

and thus, by definition,

$$f_j \notin F_k \implies \{f_j\} \neq F_k. \tag{94}$$

Furthermore, $\{f_j\}$ and F_j are not equivalent — and thus not equal — by definition:

$$|F_j| \neq 1 = |\{f_j\}| \implies F_j \neq \{f_j\} \tag{95}$$

Unintentional incorrect merges can result in laborious efforts to reconstruct the initial state. For that reason, the history all actions is tracked and individual actions can be undone or redone through the "Assignments" of the specific label source within the preference pane (section 4.1.2.1). The action history is persisted into the Paintera project and available across multiple sessions. When the data is provided as a Paintera data set, the current fragment-segment lookup table with all actions can be committed into the "fragment-segment-assignment" dataset in order to make the updated lookup table available outside the current Paintera session: Check the fragment-segment assignments box in the commit-dialog (table 13). The action history is cleared after persisting and previous actions cannot be undone anymore.

PAINTING While merging or detaching fragments is a powerful tool, in many cases that is not enough: In order to resolve undersegmentation in fragments or to create ground truth data from scratch, Paintera offers multiple tools for the direct manipulation of label data voxel values or — more intuitively — paint operations:

2D BRUSH strokes are painted into label data at the currently displayed mipmap level and at the arbitrary orientation of the 2D cross-sectionval view. The paint brush mode is active while the `space` bar is down and a circular overlay indicates the brush size that can be adjusted with the mouse-wheel. Various mouse clicks or drags trigger different actions:

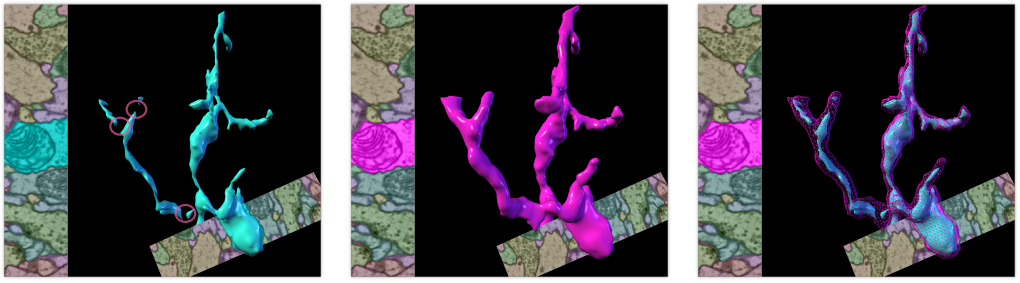
- Left click or drag paints with the currently selected label id
- Right click or drag paints transparent, *i.e.* the underlying data is uncovered.
- `shift` Right click or drag paints with the background label.

While paint brush strokes are inherently two-dimensional, it is possible to add depth or thickness to the paint brush with `shift-space` scroll.

FLOOD-FILLING overwrites the voxel values of constant regions with a new label id and is always applied at the highest resolution mipmap level. This is particular useful for splitting fragments: First, find an orientation of the 2D cross-sectional view that separates the fragment as desired. Then, apply a 2D floodfill (`F`-left click) with a new id (`N`) within this cross-section of the fragment of interest to split the fragment. Finally, apply a 3D flood-fill (`shift-F`-left click) to one side of the split fragment to ensure that one of the fragment parts is assigned a different id than the original fragment id.

Voxel manipulations are not applied to the underlying data (*background*) right away. Instead, two layers of shorter-lived layers are overlayed: a disk-cached multi-scale *canvas* with one scalar label volume per backgorund mipmap level contains a mipmap pyramid of all paint operations, and a binary mask to track the current paint operation at a specific level in the background mipmap pyramid. At any time, only one mask can be associated with a label source. Any paint operation requests a new mask and subsequent paint operations have to wait until completion of the current paint operation and the current mask is discarded. Once a paint operation is completed, the canvas at the appropriate mipmap level is overwritten with the associated label id as indicated by the mask. The changes are propagated to all mipmap levels of the canvas with up-sampling or down-sampling as appropriate and an in-memory label-block lookup keeps track of additional blocks that contain a specific label id. Finally, the current mask is discarded and new paint operations can be triggered. The canvas is considered a temporary volume and is not persisted between individual Paintera sessions. Unless committed into the background (table 13), painted data is lost when closing a Paintera session. While there is no undo operation for painting, it is possible to clear the entire canvas in the "Canvas" pane of a label source within the preference pane (section 4.1.2.1). This is a rather drastic measure and — in order to minimize potential loss of work — commiting the canvas into the background frequently is highly recommended.

LABEL MULTI-SETS In contrast to raw data, there is no obvious way to downsample label data by summarizing multiple voxels for the creation of mipmap pyramids.



(a) WTA downsampled data; ellipses highlight discontinuities.

(b) Label multisets

(c) Multiset mesh (lines, magenta) over solid WTA mesh (cyan).

Figure 53: Comparison of 3D polygon meshes generated for id 568025 of CREMI sample B at a moderate scale level of a mipmap pyramid generated with winner-takes-all downsampling (a) or with label multisets (b). While connectivity is preserved in the multiset mesh, the object is divided into multiple connected components for winner-takes-all downsampling. Moreover, the winner-takes-all meshes are subject to shrinking, which becomes apparent when rendering the multiset mesh on top (c). The resolution at the selected mipmap level is $64 \times 64 \times 80\,\mathrm{nm}^3$.

Typically, a *winner-takes-all* (WTA) scheme is employed: The downsampled voxel is the most frequent label within a small window. Winner-takes-all downsampling deletes thin or small objects. This is in particular apparent when generating 3D meshes from low-resolution winner-takes-all mipmap levels: Meshes, even for large connected components, are incomplete at relatively early levels of the mipmap pyramid (figure 53(a)). Paintera uses a non-scalar data type that summarizes voxel contents: Label *multi-sets*[42] are sorted sets of label-count pairs for a voxel. Figure 53 demonstrates the improved conservation of information in multisets compared to winner-takes-all downsampling. The sets are sorted by label to facilitate efficient containment checks through binary search. Conversion into RGB color space follows the conversion of scalar label types: The RGB representations of all contained labels are combined into a single RGB color as a weighted channel-wise sum with weights proprtional to the individual count of each label.

The distribution of synapses is a helpful guide for proof-reading segmentations. If synaptic cleft predictions are available, the subset of thresholded synapses that intersect with any selected neuron are rendered as meshes in the 3D viewer as well (figure 40). Like neuron meshes, synapse meshes are generated on-the-fly and memory-cached. Meshes are automatically updated when the synapse threshold is adjusted. At the time of writing this dissertation, this features is still experimental and not yet available through the Paintera GUI but I plan to add GUI support in the future.

[42] https://github.com/saalfeldlab/n5-label-multisets

4.1.5.5 *Conversion Helper*

The Apache Spark based `paintera-conversion-helper`[43] is a tool to conveniently convert single-scale datasets from HDF5 or N5 containers into Paintera data format. The conversion task is trivially parallizable and Apache Spark is used to multi-thread on local workstations as a command line tool or to distribute onto compute clusters. The command line tool is available through conda on the *hanslovsky* channel

```
$ conda install -c hanslovsky paintera-conversion-helper
```

and is automatically installed if Paintera is installed through conda (section 4.1.4). For submission to distributed Spark clusters, the `fat` must be compiled and the appropriate arguments passed to `spark-submit`:

```
git clone https://github.com/saalfeldlab/paintera-conversion-helper
cd paintera-conversion-helper
mvn -Pfat clean package
spark-submit \
    [spark-submit arg ...] \
    --jar=<path/to/paintera-conversion-helper-<version>.jar> \
    --class org.janelia.saalfeldlab.conversion.CommandLineConverter \
    [paintera-conversion-helper args]
```

A detailed usage example is provided in listing C.1. The help message

```
paintera-conversion-helper --help
```

provides additional information on the available parameters and options.

4.1.6 *Controls & Shortcuts*

Paintera has global controls and shortcuts for navigation and general interaction (table 11) as well as source-specific controls that are only active for specific source types (table 12 and table 13) to acommodate for data-type specific interaction requirements. Controls or shortcuts that are specific to the cross-sectional views (*CS*) or the 3D viewer (*3D*) will not be handled unless the appropriate UI element is focussed or the cursor is inside that particular UI element.

Table 11: Global navigation controls and shortcuts. The first column specifies the target UI element for each control, if any.

	Shortcut	**Action**
	P	Toggle preference pane (section 4.1.2.1)
	ctrl-O	Pop-up dataset opener context menu
	ctrl-tab	Cycle current source forward
	shift-ctrl-tab	Cycle current source backward
	V	Toggle visibility of current source
CS	M	Maximize current cross-sectional view
	Shortcut	**Action**

Table 11 — *continued on next page*

[43] https://github.com/saalfeldlab/paintera-conversion-helper

Table 11 — *continued*

	Shortcut	Action
	shift-M	Toggle single cross-sectional view with 3D viewer
	ctrl-S	Save current project state
	ctrl-shift-N	Create new label dataset
CS	left drag	Rotate cross-sectional view
CS	ctrl-left drag	Rotate cross-sectional view (slow)
CS	shift-left drag	Rotate cross-sectional view (fast)
CS	right drag	Translate within cross-sectional plane
CS	scroll	Translate perpendicular to cross-sectional plane
CS	ctrl-scroll	Translate perpendicular to cross-sectional plane (slow)
CS	shift-scroll	Translate perpendicular to cross-sectional plane (fast)
CS	ctrl-shift-scroll	Change zoom level
CS	up	Zoom in
CS	down	Zoom out
CS	left/right	Rotate left/right
CS	ctrl-left/right	Rotate left/right (slow)
CS	shift-left/right	Rotate left/right (fast)
CS	shift-Z	Axis-align
3D	left drag	Rotate data in 3D view
3D	right drag	Translate data within 3D view
3D	scroll	Move towards/away from data within 3D view
3D	shift-scroll	Move towards/away from data within 3D view (fast)
3D	right click	Open Context menu for mesh under cursor.
	Shortcut	**Action**

Table 12: Raw-data specific controls and shortcuts. The first column specifies the target UI element for each control, if any.

	Shortcut	Action
	ctrl-T	Create thresholded source from current raw source

Table 13: Label-data specific controls and shortcuts. The first column specifies the target UI element for each control, if any.

	Shortcut	Action
CS	left click	Select fragment under cursor as currently active (de-select all others)
CS	right click	Select fragment under cursor as currently active (keep others selected if any)
CS	shift-left click	Merge segments of currently active fragment and fragment under cursor
CS	shift-right click	Detach fragment under cursor from segment of currently active fragment (if the same)
CS	space-left click or drag	Paint with currently active id
CS	space-right click or drag	Erase within canvas
CS	shift-space-right click or drag	Paint background
CS	space-scroll	Change Brush Size
CS	shift-space-scroll	Change Brush Depth
CS	F-left click	2D Flood-fill at cursor with currently active id
CS	shift-F-left click	3D Flood-fill at cursor with currently active id
	N	Create new, previously unused id
	ctrl-C	Commit canvas into background and fragment-segment lookup
	C	Increment scaling factor for golden angle color mapping by one
	shift-C	Decrement scaling factor for golden angle color mapping by one
	ctrl-shift-C	Set scaling factor for golden angle color mapping
	shift-V	Toggle visibility of non-selected ids
	R	Clear mesh-caches and refresh meshes
	L	"Lock" segment for currently active fragment
	Shortcut	**Action**

4.1.7 *Extensions*

I developed Paintera as a special-purpose tool with a specific focus on neuron reconstruction for large 3D-EM. Large parts of its core functionality, however, can be re-used for processing and analysis of other kinds of volumetric data and Paintera can be extended simply by adding jars or classes with appropriate annotations and dependencies to the class path. In particular, the extensions need

to provide an implementation of `SourceState`,[44] additional entries in the open-dataset context menu for access through the user interface, and — if persistence is a requirement — serializers and deserializers for all relevant classes. In the following, I will explain how to add each of these extensions by means of a new source type that approximates the spatial gradients along each data dimension as finite differences and then computes the gradient magnitude. Project structure and imports will be omitted for better readability but the complete working example is provided in section C.1.2 of appendix C.1.

4.1.7.1 *Source State*

Instead of creating two implementations of the `SourceState` interface, common functionality of finite difference gradients and gradient magnitude features are extracted into the `Feature` interface:

```
interface Feature {
    DataSource<DoubleType, VolatileDoubleType> featureSource(
            String cacheDir,
            SourceState<? extends RealType<?>, ?>... dependsOn);
}
```

The `Feature` interfaces creates a new source from one or more data sets. A sub-directory of the `cacheDir` is used as a disk-cache for features. A `Feature` instance is passed to the `FeatureSourceState` constructor. Instead of implementing `SourceState`, it is more convenient to extend the `MinimalSourceState` in many cases:

```
public class FeatureSourceState extends MinimalSourceState<
        DoubleType,
        VolatileDoubleType,
        DataSource<DoubleType, VolatileDoubleType>,
        ARGBColorConverter<VolatileDoubleType>> {

    private final Feature feature;

    public FeatureSourceState(
            final Feature feature,
            final String name,
            final String cacheDir,
            SourceState<? extends RealType<?>, ?>... dependsOn) {
        super(
                feature.featureSource(cacheDir, dependsOn),
                new ARGBColorConverter.InvertingImp1<VolatileDoubleType>(),
                new ARGBCompositeAlphaAdd(),
                name,
                dependsOn);
        this.feature = feature;
        converter().setMin(0.0);
        converter().setMax(50.0);
    }

}
```

The generic arguments for the `MinimalSourceState` specify the type of the data (in this case, `DoubleType`), the type of the data that is passed to the viewer (in practice, volatile types like `VolatileDoubleType` are used), the source that manages data loading (`DataSource<DoubleType, VolatileDoubleType>`), and a converter

[44] `org.janelia.saalfeldlab.paintera.state.SourceState`

to map converted viewer data into ARGB space. It will be necessary to keep the `Feature` instance as a member variable for serialization. The converter is arbitrarily pre-set to a reasonable value range for CREMI sample B. Overriding the `SourceState.onAdd` ensures that the cross-sectional views are being re-rendered on changes of the converter settings:

```
@Override
public void onAdd(PainteraBaseView paintera) {
    InvalidationListener reqRep = obs -> paintera
            .orthogonalViews()
            .requestRepaint();
    converter().minProperty().addListener(reqRep);
    converter().maxProperty().addListener(reqRep);
    converter().colorProperty().addListener(reqRep);
    converter().alphaProperty().addListener(reqRep);
}
```

The majority of the code lives in the individual implementations of the `Feature` interface, `GradientFeature` and `MagnitudeFeature`, respectively. As the implementation detail is more of an exercise in ImgLib2[45] and not particularly relevant for the demonstration of Paintera extensions, I refer the reader to section C.1.2 for the specific implementations and the ImgLib2 tutorials if a deeper understanding is desired.

4.1.7.2 *Context Menu Entry*

The open dataset context menu can be extended with additional entries by implementing the `OpenDialogMenuEntry` and annotating the implementing class as a SciJava `Plugin`:

```
@Plugin(type = OpenDialogMenuEntry.class, menuPath = "_Features>_Gradient Magnitude")
class MenuEntry implements OpenDialogMenuEntry
```

The `menuPath` annotation parameter specifies the location inside the open dataset menu tree: `>` indicates sub-menus, and underscores before a character create mnemonics for accelerated access when the appropriate key is pressed. The single interface method

```
BiConsumer<PainteraBaseView, String> onAction();
```

defines the behavior of the new menu entry. The returned `BiConsumer` takes the Paintera viewer class and the project directory as input. In the example implementation, the return value is implemented as a Java8 lambda expression:[46]

```
return (pbv, directory) -> {
 // implementation goes here
}
```

First, all available real-valued data sources are extracted from the viewer

```
final Predicate<SourceState<?, ?>> isReal =
        state -> state
                .getDataSource()
                .getDataType() instanceof RealType<?>;
final List<SourceState<? extends RealType<?>, ?>> sources =
        pbv
```

[45] https://imagej.net/ImgLib2

[46] https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html

```
        .sourceInfo()
        .trackSources()
        .stream()
        .map(pbv.sourceInfo()::getState)
        .filter(isReal)
        .map(s -> (SourceState<? extends RealType<?>, ?>)s)
        .collect(Collectors.toList());
```

Next, a JavaFX dialog is created with UI elements to prompt the user to choose from the available real-valued data sources:

```
final Alert alert = PainteraAlerts.alert(
        Alert.AlertType.CONFIRMATION,
        true);
final ObservableList<SourceState<? extends RealType<?>, ?>> choices =
        FXCollections.observableArrayList(sources);
final ComboBox<SourceState<? extends RealType<?>, ?>> comboBox =
        new ComboBox<>(choices);
alert.getDialogPane().setContent(comboBox);
final Optional<ButtonType> bt = alert.showAndWait();
```

If the user made a valid selection and clicked the “OK” button,

```
if (bt.filter(ButtonType.OK::equals).isPresent()
        && comboBox.getValue() != null)
```

first, partial derivatives along each data dimension are created:

```
final SourceState<? extends RealType<?>, ?> raw = comboBox.getValue();
final int nDim = raw.getDataSource().getDataSource(0, 0).numDimensions();
final FeatureSourceState[] gradients = IntStream
        .range(0, nDim)
        .mapToObj(dim -> new GradientFeature(dim))
        .map(feat -> new FeatureSourceState(
                feat,
                raw.nameProperty().getName() + "-gradient", directory, raw))
        .toArray(FeatureSourceState[]::new);
```

Second, the gradient magnitude is calculated from the partial derivatives:

```
final FeatureSourceState magnitude = new FeatureSourceState(
        new MagnitudeFeature(),
        raw.nameProperty().getName() + "-gradient-magnitude",
        directory,
        gradients);
```

Third, converters are set up to map the partial derivatives to red, green, and blue color, respectively:

```
gradients[0]
        .converter()
        .setColor(Colors.toARGBType("#ff0000"));
gradients[1]
        .converter()
        .setColor(Colors.toARGBType("#00ff00"));
gradients[2]
        .converter()
        .setColor(Colors.toARGBType("#0000ff"));
```

And, finally, the partial derivatives and gradient magnitude are added to the viewer:

```
Stream.of(gradients).forEach(pbv::addState);
Stream.of(magnitude).forEach(pbv::addState);
```

It is now possible to create gradient features for arbitrary real-valued data sources. If serialization of the Paintera project with gradient features is required, JSON serializers and deserializers have to be provided as described in the next section.

4.1.7.3 *Serialization*

The *Gson*[47] Java library is used for serialization into and deserialization from JSON strings. Gson (de-)serializers need to be added for all added classes that cannot be automatically (de-)serialized by Gson. In order to auto-detect these (de-)serializers and register them with Gson instances, Paintera extends extends the `JsonSerializer` and `JsonDeserializer` interfaces of the Gson library to specify the target class for serialization or deserialization:

```
interface ScijavaUtils.HasTargetClass<T> {
    Class<T> getTargetClass();
}
interface PainteraSerialization.PainteraDeserializer<T> extends
        JsonDeserializer<T>,
        SciJavaPlugin,
        SciJavaUtils.HasTargetClass<T> {
    default boolean isHierarchyAdapter() {
        return false;
    }
}

interface PainteraSerialization.PainteraSerializer<T> extends
        JsonSerializer<T>,
        SciJavaPlugin,
        SciJavaUtils.HasTargetClass<T> {
    default boolean isHierarchyAdapter() {
        return false;
    }
}

public interface PainteraSerialization.PainteraAdapter<T> extends
        PainteraSerializer<T>,
        PainteraDeserializer<T>,
        SciJavaUtils.HasTargetClass<T> {
    default boolean isHierarchyAdapter() {
        return false;
    }
}
```

If the optional `isHierarchyAdapter` method returns `true`, the serializer, deserializer, or adapter is registered as hierarchy adapter. Implementations of these interfaces need to be annotated as SciJava `Plugin` appropriately:

```
@Plugin(type = PainteraSerialization.PainteraSerializer.class)
class Serializer
implements PainteraSerialization.PainteraSerializer {}
@Plugin(type = PainteraSerialization.PainteraDeserializer.class)
class Serializer implements
PainteraSerialization.PainteraDeserializer {}
@Plugin(type = PainteraSerialization.PainteraAdapter.class)
class Adapter implements
PainteraSerialization.PainteraAdapter {}
```

In some cases — *e.g.* this example — the viewer state and project directory are needed for (de-)serialization and (de-)serializers or adapters cannot be instantiated before the viewer is initialized. Instead, factories can be provided to create stateful (de-)serializers and adapters:

```
interface StatefulSerializer.SerializerFactory<
        T,
```

[47] https://github.com/google/gson

```
        S extends JsonSerializer<T>> extends
        SciJavaPlugin,
        SciJavaUtils.HasTargetClass<T> {
    S createSerializer(
            Supplier<String> projectDirectory,
            ToIntFunction<SourceState<?, ?>> stateToIndex);
}

interface StatefulSerializer.DeserializerFactory<
        T,
        S extends JsonDeserializer<T>> extends
        SciJavaPlugin,
        SciJavaUtils.HasTargetClass<T> {
    S createDeserializer(
            Arguments arguments,
            Supplier<String> projectDirectory,
            IntFunction<SourceState<?, ?>> dependencyFromIndex);
}

interface StatefulSerializer.SerializerAndDeserializer<
        T,
        D extends JsonDeserializer<T>,
        S extends JsonSerializer<T>> extends
        SerializerFactory<T, S>,
        DeserializerFactory<T, D>,
        SciJavaPlugin,
        SciJavaUtils.HasTargetClass<T> {}
```

Similar to the annotations of stateless (de-)serializers, the factories need to be annotated for automatic detection and registration:

```
@Plugin(type = StatefulSerializer.SerializerFactory.class)
class Serializer
implements StatefulSerializer.SerializerFactory {}
@Plugin(type = StatefulSerializer.DeserializerFactory.class)
class Serializer implements
StatefulSerializer.DeserializerFactory {}
@Plugin(type = StatefulSerializer.SerializerAndDeserializer.class)
class SerializerAndDeserializer implements
StatefulSerializer.SerializerAndDeserializer {}
```

The source in this gradient feature example depends on other sources that are only available when the viewer is initialized and populated with the dependencies, and thus stateful (de-)serializers are provided by a `StatefulSerializer.SerializerAndDeserializer` factory. The serializer

```
class Serializer implements JsonSerializer<FeatureSourceState> {

    private final ToIntFunction<SourceState<?, ?>> sourceToIndex;

    private Serializer(final ToIntFunction<SourceState<?, ?>> sourceToIndex) {
        this.sourceToIndex = sourceToIndex;
    }

    @Override
    public JsonElement serialize(
            FeatureSourceState src,
            Type typeOfSrc,
            JsonSerializationContext context) {
        final JsonObject map = new JsonObject();
        // populate map
        return map;
    }
}
```

overrides the `serialize` method to return a JSON object that holds a list of dependencies identified by their respective indicies,

```
map.add(DEPENDS_ON_KEY, context.serialize(Stream
        .of(src.dependsOn())
        .mapToInt(sourceToIndex)
        .toArray()));
```

the feature instance,

```
map.add("feature", serializeWithClassInfo(
        src.feature,
        context));
```

and user-specified settings of the source, *e.g.* converter or composition mode:

```
map.add("composite", serializeWithClassInfo(
        src.compositeProperty().get(),
        context));
map.add("converter", serializeWithClassInfo(
        src.converter(),
        context));
map.add(INTERPOLATION_KEY, context.serialize(
        src.interpolationProperty().get(),
        Interpolation.class));
map.addProperty(
        IS_VISIBLE_KEY,
        src.isVisibleProperty().get());
map.addProperty(NAME_KEY, src.nameProperty().get());
```

Similarly, the deserializer

```
class Deserializer implements JsonDeserializer<FeatureSourceState> {
    private final IntFunction<SourceState<?, ?>> dependencyFromIndex;

    private final String cacheDir;

    private Deserializer(
            final IntFunction<SourceState<?, ?>> dependencyFromIndex,
            final String cacheDir) {
        this.dependencyFromIndex = dependencyFromIndex;
        this.cacheDir = cacheDir;
    }

    @Override
    public FeatureSourceState deserialize(
            JsonElement json,
            Type typeOfT,
            JsonDeserializationContext context) throws JsonParseException {
        final JsonObject map = json.getAsJsonObject();
        try {
            // instantiate and return state from map
        } catch (ClassNotFoundException e) {
            throw new JsonParseException(e);
        }
    }
}
```

overrides the `deserialize` method to ensure that all dependencies were deserialized,

```
final SourceState<? extends RealType<?>, ?>[] dependsOn
                    = IntStream
                    .of(context.deserialize(
                            map.get(DEPENDS_ON_KEY),
                            int[].class))
                    .mapToObj(dependencyFromIndex)
```

```
                    .toArray(SourceState[]::new);
if (Stream.of(dependsOn).anyMatch(Objects::isNull))
    return null;
```

then deserializes the feature instance and instantiates the source state,

```
final FeatureSourceState fs = new FeatureSourceState(
        deserializeFromClassInfo(
                map.getAsJsonObject("feature"),
                context),
        map.get(NAME_KEY).getAsString(),
        cacheDir,
        dependsOn);
```

and finally updates the state with serialized settings and returns the state:

```
final ARGBColorConverter<VolatileDoubleType> converter =
        deserializeFromClassInfo(
                map.getAsJsonObject("converter"),
                context);
fs.converter().setColor(converter.getColor());
fs.converter().setMin(converter.getMin());
fs.converter().setMax(converter.getMax());
fs.converter().alphaProperty().set(converter.alphaProperty().get());
fs.compositeProperty().set(deserializeFromClassInfo(
        map.getAsJsonObject("composite"),
        context));
fs.interpolationProperty().set(context.deserialize(
        map.get(INTERPOLATION_KEY),
        Interpolation.class));
fs.isVisibleProperty().set(map.get(IS_VISIBLE_KEY).getAsBoolean());
return fs;
```

The `SerializationFactory` class creates stateful (de-)serializes and appropriate annotation ensures that the factory is detected automatically:

```
@Plugin(type = StatefulSerializer.SerializerAndDeserializer.class)
class SerializationFactory implements
        StatefulSerializer.SerializerAndDeserializer<
        FeatureSourceState,
        Deserializer,
        Serializer> {

    @Override
    public Deserializer createDeserializer(
            StatefulSerializer.Arguments arguments,
            Supplier<String> projectDirectory,
            IntFunction<SourceState<?, ?>> dependencyFromIndex) {
        return new Deserializer(
                dependencyFromIndex,
                projectDirectory.get());
    }

    @Override
    public Serializer createSerializer(
            Supplier<String> projectDirectory,
            ToIntFunction<SourceState<?, ?>> stateToIndex) {
        return new Serializer(stateToIndex);
    }

    @Override
    public Class<FeatureSourceState> getTargetClass() {
        return FeatureSourceState.class;
    }
}
```

The extension is now complete with support for serialization. The complete example is provided as a Maven project with installation and execution instructions in appendix B.2.

4.1.8 *Interactive Agglomeration*

One of the intended use-cases for Paintera is an optimized proof-reading workflow that utilizes user annotations to train an agglomeration model online: Starting from an existing segmentation, expert users provide examples for pairs of fragments that should be merged or split. A machine-learning algorithm learns the parameters of an agglomeration model from these annotations and an updated agglomeration that respects the annotations as constraints is presented to the users. This workflow can be used to rapidly proof-read automatic neuron reconstructions for use in biological analysis or the rapid creation of vast amounts of training data.

Multi-cuts have been used succesfully for agglomeration of fragments in the context of neuron reconstruction. Given a graph $\mathcal{G} = (V, E, w)$ with N vertices $V = \{v \in \mathbb{N}^+ : v \leq N\}$, edges $E \subset V \times V$, and weights $w \in \mathbb{R}^{|E|}$ associated with each edge $e \in E$, the multi-cut objective

$$y^* = \arg\min_{y \in \{0,1\}^{|E|}} \sum_{i=1}^{|E|} w_i y_i \tag{96}$$

$$\text{subject to} \quad \forall C \in \text{cycles}(\mathcal{G}) \; \forall e \in C : \; y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{97}$$

partitions the vertices into clusters, where $y_i = 1$ means that an edge $e \in E$ is "cut", *i.e.* e_1 and e_2 are not in the same cluster. Positive weights are attractive — *i.e.* two vertices are more likely to be part of the same cluster; negative weights are repulsive — two vertices are more likely to be in different clusters. The multi-cut constraints prevent so-called dangling edges: For a cut edge $e \in E$ there is a path of uncut edges that connects e_1 and e_2.

A binary random forest classifier is trained from user annotations to predict the weights w of the multi-cut model. Two annotations at the minimum — one per each class — are required for training the random forest. Typically, random forests achieve good accuracy with only few and sparse annotations, and users should expect reasonable — but not perfect — results within minutes of proof-reading work.

Optimization of the multi-cut objective is NP-Hard (Bansal, Blum, and Chawla 2004) and solving for optimality usually requires commercial integer linear programming (ILP) frameworks like CPLEX[48] or Gurobi.[49] The high cost of licenses for these frameworks can be prohibitive for users who do not have access to free or cheaper academic licenses. The C++ library OpenGM (Bjoern Andres, Beier, and Kappes 2012) for discrete graphical models implements — besides ILP optimization with CPLEX or Gurobi — approximate solvers that are both reasonably

[48] https://www.ibm.com/analytics/cplex-optimizer

[49] http://www.gurobi.com

fast and accurate for the use of multi-cut agglomeration in neuron reconstruction. Beier et al. (2017) used the C++ library for graph-based image segmentation nifty[50] (Beier 2018) as an alternative solver for the multi-cut model. Unfortunately, Java can access native code — such as compiled C or C++ — only through the Java Native Interface (JNI), which is cumbersome at best. Instead, I decided to build a client-server architecture with ZeroMQ[51] based communication: User annotations are sent from Paintera-based client (section 4.1.8.1) to a server (section 4.1.8.1) that trains a predictor for the agglomeration parameters from user annotations, optimizes the multi-cut objective with updated parameters, and broadcasts the agglomeration result back to the client.

4.1.8.1 *Paintera Interactive Solver Server*

The Paintera Interactive Solver Server (pias)[52] is implemented in Python: High-level language features reduce boiler-plate code and both nifty and ZeroMQ are readily available through extensive Python bindings.

Pias requires Python 3.6 or more recent and most dependencies are available through PyPI and installed automatically when using pip. The nifty and z5py[53] Python packages are not available through the Python Package Index (PyPI)[54] and have to be built from source (follow instructions on the respective GitHub repositories) or can be installed through conda through the `cpape` and `conda-forge` channels, respectively:

```
conda install -c cpape nifty
conda install -c conda-forge z5py
```

If the `pyzmq` dependency is not installed through conda, make sure that `libzmq` https://github.com/zeromq/libzmq is installed on your system.

Finally, pias is available on PyPI for installation with pip:

```
pip install pias
```

Alternatively, the latest development version can be installed directly from GitHub:

```
pip install --user git+https://github.com/saalfeldlab/pias
```

This installs the piac Python library and the piac command:

```
pias --help
usage: pias [-h] --container CONTAINER --paintera-dataset PAINTERA_DATASET
            [--directory DIRECTORY] [--num-io-threads NUM_IO_THREADS]
            [--log-level {NOTSET,DEBUG,INFO,WARN,ERROR,CRITICAL,FATAL,TRACE}]
            [--version]

optional arguments:
  -h, --help            show this help message and exit
  --container CONTAINER
                        N5 FS Container with group that contains edges as
                        pairs of fragment labels and features
  --paintera-dataset PAINTERA_DATASET
                        Paintera dataset inside CONTAINER that also contains
```

[50] https://github.com/DerThorsten/nifty
[51] http://zeromq.org
[52] https://github.com/saalfeldlab/pias
[53] https://github.com/constantinpape/z5
[54] https://pypi.org

```
                        datasets `edges' and `edge-features'
  --directory DIRECTORY
                        Directory for ipc sockets and serialization of server
                        state.
  --num-io-threads NUM_IO_THREADS
  --log-level {NOTSET,DEBUG,INFO,WARN,ERROR,CRITICAL,FATAL,TRACE}
  --version             show program's version number and exit
```

The container is the path to a filesystem N5 container that contains a paintera dataset (section 4.1.5.2) with additional datasets "edges" and "edge-features". "edges" is a two-dimensional, uint64 dataset with dimensions $2 \times n_e$, where n_e is the number of pairwise edges between pairs of neighboring fragments. Each row of the list represents an edge identified by the pair of label ids $(id_1, id_2) : id_1 < id_2$ of the fragments. "edge-features" is a two-dimensional `float64` dataset with dimensions $n_c \times n_e$, where n_c is the number of feature channels. Each row in the "edge-features" dataset is the feature vector for the edge identified by the corresponding row in the "edges" dataset. It is the user's responsibility to create those datasets (and update them if necessary) to follow these specifications. The server will use this directory to store temporary data and to create multiple ZeroMQ inter-process transport (ipc) sockets for communication with the client — note that all messages are sent in big endian order (all paths relative to the server directory):

SERVER The `server` reply (REP) socket can be used to request general information about the server through multiple endpoints. The `/help` endpoint sends a single zmq string response. All endpoints under `/api` send at least two messages:

1. a single integer

   0 endpoint is known

   1 unknown error during processing

   2 endpoint is unknown

2. a single integer specifying the number of messages to be sent (may be zero)

3. optional, if number of messages is larger than zero, n times:

   - integer indicating type of message

     0 string

     1 bytes

     2 integer

     3 unknown/structured (look at help if available, will be sent as bytes)

   - actual contents

These endpoints are available:

`/` Ping the server with an arbitrary message (possibly empty). The pong response is sent as an empty string message.

`/help` Request a help message.

`/api/n5/container` Path to n5 container holding paintera dataset with edges and features

`/api/n5/dataset` Path to paintera dataset in n5 container

`/api/n5/all` Send both container and dataset as multiple messages

`/api/save-ground-truth-labels` Serialize current ground truth labels (uv-pairs and labels) into server directory

SERVER-PING The server can be pinged through the `server-ping` REP socket with an arbitrary message (possibly empty). The pong response is sent as an empty string message.

SERVER-CURRENT-SOLUTION The latest valid solution of the agglomeration model can be requested with an arbitrary message to the `server-current-solution` REP socket. The response is an array of uint64 with segment assignments for all fragments.

SERVER-SET-EDGE-LABELS Submit ground-truth edge annotations to the server as an array of a sorted pair of two uint64 fragment ids identifying the edge and a single int32 edge label: 0 for split, 1 for merge.

SERVER-UPDATE-SOLUTION Request an update of the agglomeration model solution: First, a random forest classifier is trained with the current annotated edge labels, then the multi-cut model with predicted edge-weights is optimized to update the agglomeration.

4.1.8.2 *Paintera Interactive Solver Client*

Building on top of the existing support for label data in Paintera, the Paintera Interactive Solver Client (piac)[55] is implemented in the Kotlin programming language[56] as a Paintera extension (section 4.1.7). In order to communicate with the server-side zmq sockets, piac uses the jzmq[57] library that wraps native zmq for use in Java through the JNI. Before installing piac, make sure that the native zmq library is installed and compile the jzmq JNI bindings:

```
git clone https://github.com/zeromq/jzmq
cd jzmq/jzmq-jni
./autogen.sh
./configure
make
make install
```

Currently, piac has not been released yet and has to be installed from source:

```
git clone https://github.com/saalfeldlab/piac
cd piac
mvn clean install
```

[55] https://github.com/saalfeldlab/piac
[56] https://kotlinlang.org
[57] https://github.com/zeromq/jzmq

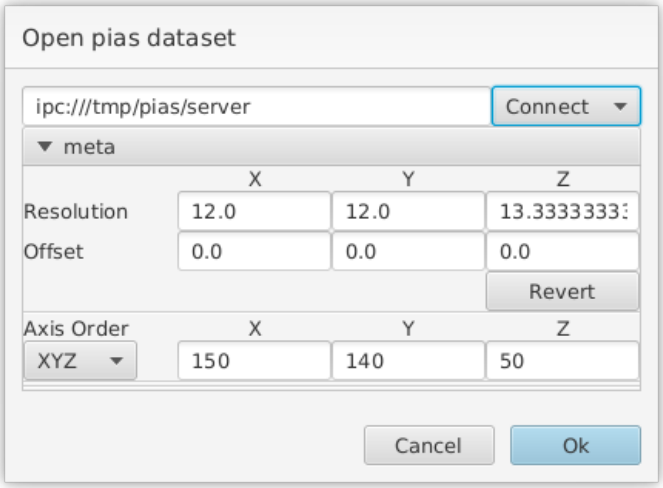


Figure 54: Connect to a pias instance: The server URL is composed of protocol (`ipc://`), the path to the server directory (`/tmp/pias/` in this example), and the `server` socket. The server provides N5 container location and dataset for the label dataset. Meta data are read from the dataset attributes but can be adjusted before the dataset is added after confirmation.

Then, start Paintera with support for interactive agglomeration with

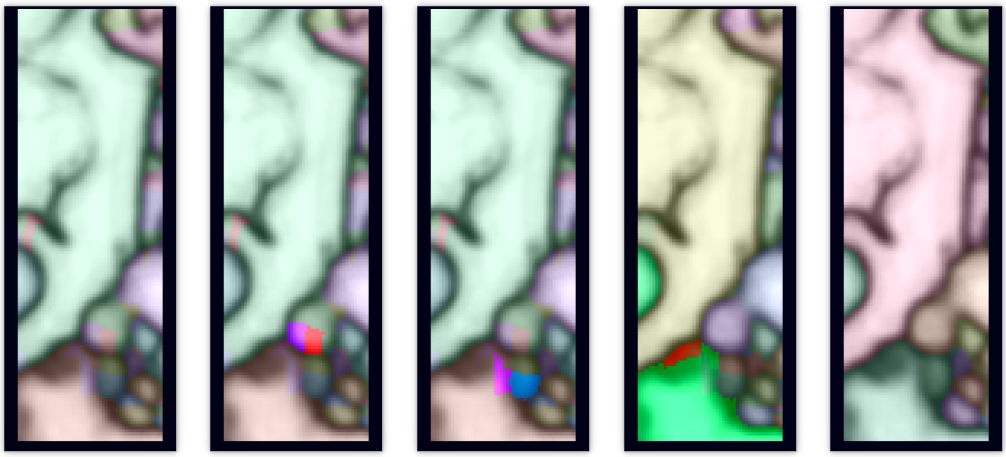
```
paintera \
    --additional-endpoints \
    org.janelia.saalfeldlab:piac:0.1.0-SNAPSHOT \
    --
```

The open dataset context menu will show an additional "Pias" item that toggles a dialog to connect to a pias instance (figure 54). On confirmation, a label dataset is added to Paintera. Following the controls for label data, examples of merge and split edges are provided to the pias instance with mouse clicks with an active fragment: Shift-left click creates a merge example, shift-right click indicates a split (figure 55).

4.2 EQIP

The Python machine learning community has seen a wealth of frameworks for general machine learning (Pedregosa et al. 2011) and deep learning (Bergstra et al. 2010; Jia et al. 2014; Abadi et al. 2016) emerge over the past decade. The *Gunpowder*[58] framework was created to facilitate machine learning workflows for volumetric, multi-dimensional images. Gunpowder is developed and maintained at Janelia and is used mostly in the context of electron micrographs of nervous systems, and it was thus a logical decision to implement and train the quasi-isotropic network architecture with Gunpowder and TensorFlow as a deep learning backend. While Gunpowder greatly simplifies the construction of machine learning workflows, it is not concerned with reproducibility: the same experiment, for example, could be executed twice with different Python interpreters

[58] https://github.com/funkey/gunpowder



(a) Initial state (b) Merge (c) Split (d) Merge (e) Updated state

Figure 55: Providing merge and split examples to a pias instance generates reasonable agglomerations with only a few mouse clicks: In the initial state (a), fragments are not agglomerated. After providing one example for merge (b) and split (c) edges, the pias instance can train a predictor for the agglomeration model parameters and update the agglomeration (d). Additional examples (d) help refine the agglomeration (e).

that use different versions of required packages and, consequently, results may differ as bug fixes or new features change the behavior of individual packages. Preferrably, an experiment reproduces the same result every time it is executed. A computational experiment should not be corrected by changing its internal state but rather by making a copy that contains the necessary updates of code and dependencies. I created a collection of *Electron-microscopy Quasi-Isotropic Prediction*[59] scripts to construct an environment of reproducible experiments for the training of quasi-isotropic networks as described in chapter 3 with different parameters and losses. Training and prediction workflows build on Gunpowder with additional specialized nodes for quasi-isotropic networks that are available in the Gunpowder extension package fuse.[60] Furthermore, EQIP uses Daisy[61] to make use all local GPUs of a GPU server for distributed prediction.

4.2.1 *Architecture*

Reproducibility of an experiment requires that changeable parameters such as software dependencies or program options and model parameters are encapsulated in an unmodifiable entity. Initially, I used Docker containers (Merkel 2014) to manage software versions and text files to store experiment parameters. Eventually, the additional effort for maintenance and debugging when using Docker images and — more importantely — the lack of TensorFlow Docker images with

[59] https://github.com/saalfeldlab/eqip
[60] https://github.com/saalfeldlab/fuse
[61] https://github.com/funkelab/daisy

support for Python3.6[62] led to the decision to use conda[63] instead of Docker for managing software. While conda does not offer a separation that is as strict as Docker containers, it is sufficient for the use case of neural network training, in particular in an environment without cluster management software. The availability of Python versions 3.6 and more recent and TensorFlow with GPU through conda packages ensure efficient use of graphics cards and recent updates of Python.

Experiment management is simple: First, an *experiment* is created as a directory that contains a conda environment with EQIP and all its dependencies, a sub-directory for all the data used in the experiment, a Python script to create a new *setup* for this experiment, and scripts for prediction and further processing as needed. A setup is an instantiation of the experiment with specific parameters, such as the loss or the number of and range of affinities in neuron-reconstruction experiments. Setup data is stored in a sub-directory of the experiment with alpha-numeric name counting up as more setups are created. The setup directory contains a conda environment, separate shell scripts to create the network architecture and to initiate or re-start training, and additional files for meta-data if necessary. By default, a setup inherits the experiment's conda environment but additional packages can be specified as needed. It is best practice, however, to create a new experiment if additional packages or updates introduce major changes.

Accidental changes to experiment or setup are prevented by setting read-only — and executable as needed — flags for all text files and shell or Python scripts. In order to enable byte-code caching of Python packages or use of other temporary files, the conda environments are left modifiable. While this is a risk for unintended modification of these conda environments, this is highly unlikely to occur in practice, as the experiment and setup environments are stored in the appropriate sub-directories of the experiment directory tree.

4.2.2 *Installation & Use*

EQIP can be installed through the package installer for Python (pip):

```
# install the latest development version:
$ pip install git+https://github.com/saalfeldlab/eqip
# install release version 0.5.1
$ pip install git+https://github.com/saalfeldlab/eqip@0.5.1
```

Currently, EQIP has to be installed from GitHub directly as some dependencies are not (yet) available on the *Python Package Index* (PyPI). Specific versioned releases can be installed through the `@<version>` syntax.

4.2.2.1 *Creation of Quasi-Isotropic Experiments and Setups*

The `create-experiment` command is installed with EQIP:

```
$ create-experiment EXPERIMENT_TYPE [OPTIONS]
$ create-experiment -h
```

[62] The operating system on TensorFlow Docker images is Ubuntu 16.04, which only provides Python 3.5: https://github.com/tensorflow/tensorflow/issues/22292

[63] https://conda.io

```
usage: create-experiment [-h] {affinities-with-glia}

positional arguments:
  {affinities-with-glia}

optional arguments:
  -h, --help            show this help message and exit
```

Currently, only the `affinities-with-glia` (section 3.4) experiment type is available. The experiment options are used to specify the experiment directory, data, and EQIP revision, for example for the `affinities-with-glia` experiment:

```
$ create-experiment -- affinities-with-glia --help
usage: create-experiment [-h] [--experiment-name EXPERIMENT_NAME]
                         --data-pattern PATTERN [--copy-data] [--overwrite]
                         [--conda-sh CONDA_SH] [--eqip-revision EQIP_REVISION]
                         [--log-level {DEBUG,INFO,WARN,ERROR,CRITICAL}]
                         PATH

positional arguments:
  PATH                  Path to experiment directory

optional arguments:
  -h, --help            show this help message and exit
  --experiment-name EXPERIMENT_NAME
                        Defaults to basename of PATH.
  --data-pattern PATTERN
                        Glob pattern specifying the data to be used. Files are
                        sym-linked by default.
  --copy-data           Copy data instead of sym-linking.
  --overwrite           Overwrite experiment if it already exists
  --conda-sh CONDA_SH   Path to conda.sh
  --eqip-revision EQIP_REVISION
                        EQIP revision to use in experiment. Defaults to latest
                        master
  --log-level {DEBUG,INFO,WARN,ERROR,CRITICAL}
                        Set log level for experiment creation.
```

The positional `PATH` argument specifies the experiment directory. Experiment creation fails if the experiment directory already exists and the `ptin` is not set. The experiment name is inferred from the base path of the experiment directory but can be set explicitly through the `--experiment-name` option. Files or directories that match the glob pattern specified by `--data-pattern` are linked symbolically or copied with the `--copy-data` flag into the `data` sub-directory inside the experiment directory. Conda is used for dependency management and `conda.sh` activates the experiment and setup environments in the respective scripts. If conda is not installed in a default location or a different conda installation is preferred, the path to `conda.sh` can be specified through the `--conda-sh` option. The `--eqip-revision` option sets the EQIP revision for the experiment. By default, this is the version of the installed EQIP package or the current master branch if not on a release version. The `--log-level` sets the logging verbosity for experiment creation and is helpful for debugging experiment creation. Setups can be created with the `create-setup.py` script inside the experiment directory (`PATH`):

```
$ cd $PATH
$ conda activate $PWD/conda-env
$ ./create-setup.py --help
usage: create-setup.py [-h] --affinity-neighborhood-x AFFINITY_NEIGHBORHOOD_X
                       [AFFINITY_NEIGHBORHOOD_X ...] --affinity-neighborhood-y
                       AFFINITY_NEIGHBORHOOD_Y [AFFINITY_NEIGHBORHOOD_Y ...]
```

```
                      --affinity-neighborhood-z AFFINITY_NEIGHBORHOOD_Z
                      [AFFINITY_NEIGHBORHOOD_Z ...] --mse-iterations
                      MSE_ITERATIONS --malis-iterations MALIS_ITERATIONS
                      [--data-provider DATA_PROVIDER]
                      [--log-level {DEBUG,INFO,WARN,ERROR,CRITICAL}]
                      [--additional-pip-packages ADDITIONAL_PIP_PACKAGES
                                                 [ADDITIONAL_PIP_PACKAGES ...]]

Create new setup for experiment `test-experiment'. Unknown arguments will be
collected and passed to `train-affinities-on-interpolated-ground-truth-with-
glia' as additional arguments. See `train-affinities-on-interpolated-ground-
truth-with-glia --help' for details and avoid duplicate arguments.

optional arguments:
  -h, --help            show this help message and exit
  --affinity-neighborhood-x AFFINITY_NEIGHBORHOOD_X [AFFINITY_NEIGHBORHOOD_X ...]
                        Affinity ranges for x-axis of data.
  --affinity-neighborhood-y AFFINITY_NEIGHBORHOOD_Y [AFFINITY_NEIGHBORHOOD_Y ...]
                        Affinity ranges for y-axis of data.
  --affinity-neighborhood-z AFFINITY_NEIGHBORHOOD_Z [AFFINITY_NEIGHBORHOOD_Z ...]
                        Affinity ranges for z-axis of data.
  --mse-iterations MSE_ITERATIONS
                        Number of iterations of training with mean squared
                        error loss.
  --malis-iterations MALIS_ITERATIONS
                        Number of iterations of training with malis loss.
  --data-provider DATA_PROVIDER
                        Specify data providers if other than
                        EXPERIMENT_DIR/data/*
  --log-level {DEBUG,INFO,WARN,ERROR,CRITICAL}
                        Set log level for setup creation.
  --additional-pip-packages ADDITIONAL_PIP_PACKAGES [ADDITIONAL_PIP_PACKAGES ...]
                        Install additional packages through pip. Experiment
                        conda environment will be cloned instead of sym-
                        linked.
```

The `create-setup.py` script will create a new sub-directory with the smallest unused integral number starting at zero as name. The setup directory contains the setup-specific conda environment at `conda-env` that is a sym-link to the experiment conda environment unless additional Python packages are installed through the `--additional-pip-packages` flag — then, a clone of the experiment conda environment is updated with the new packages. The `offsets` text file specifies the connectivity of the affintiy graph in units of voxels along each dimension as specified by the various `affinity-neighborhood-{x,y,z}` options. Affinities are symmetric and therefore it is sufficient to specify and train only half of the connections per voxel. By convention, neighborhoods are specified as negative numbers but positive numbers can be used as well. A network architecture for nearest neighbor affinities only, for example, would by specified by setting each affinity neighborhood to -1. Quasi-isotropic networks can be trained with a combination of mean squared error (MSE) loss followed by malis loss (Turaga, Kevin L Briggman, et al. 2009). The number of iterations for each MSE and malis can be specified with the `--mse-iterations` and the `--malis-iterations` options, respectively, and either can be set to zero. Unmatched arguments are passed as additional arguments for training. [64]. Finally, the `mknet.sh` and `train.sh` should be run in succession, to create the network architecture and initiate or re-start

[64] `train-affinities-on-interpolated-ground-truth-with-glia --help` lists available training options.

training, respectively. `train.sh` expects the index of the GPU that is selected for training as single positional argument. Then, creating an experiment and a setup from scratch and initiating training is as simple as:

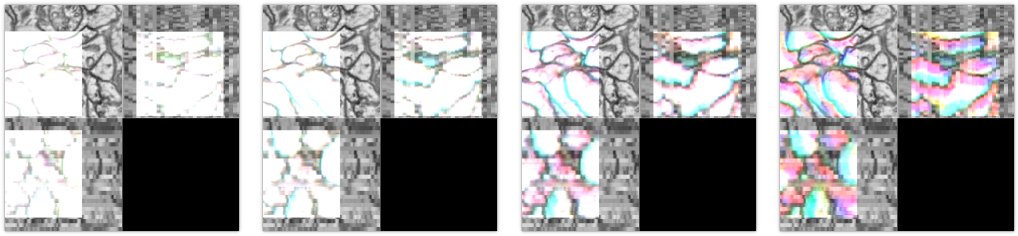
```
create-experiment \
    affinities-with-glia \
    $HOME/my-experiment \
    --data-pattern='/data/labeled/sample_*.n5'
cd $HOME/my-experiment
conda activate $PWD/conda-env
./create-setup.py \
    --affinity-neighborhood-x -1 -2 -5 -10 \
    --affinity-neighborhood-y -1 -2 -5 -10 \
    --affinity-neighborhood-z -1 -2 -5 -10 \
    --mse-iterations=400000 \
    --malis-iterations=0 \
    --pre-cache-num-workers=10 \
    --pre-cache-size=20 \
    --ignore-labels-for-slip \
    --save-checkpoint-every=2000 \
    --snapshot-every=250
cd 0
./mknet.sh
./train.sh 0
```

This creates a network architecture for an affinity graph with axis-aligned affinity offsets with steps $(-1, -2, -5, -10)$ along each data dimension. The network is trained with MSE loss for 400,000 iterations and zero iterations for Malis loss. Pre-caching 20 samples with 10 parallel CPU workers minimizes GPU idle time during expensive data-loading and augmentation. Single-section misalignment augmentations are not applied to label data with the `--ignore-labels-for-slip` option; misalignment augmentations that affect larger parts of a section series will still be considered. Model checkpoints are saved every 2000 iterations to allow to resume training in case of errors or intended interruptions. Snapshots of relevant data help monitor training progress (section 4.2.2.2) beyond the training loss and are created as HDF5 files every 500 training iterations. The `mknet.sh` creates the network architecture and `train.sh` starts training on the first GPU.

4.2.2.2 *Monitoring Experiments*

Typically, the training loss is evaluated to monitor the progress of an experiment, *e.g.* TensorBoard visualizes training loss as a function of iterations or wall time. It is not unusual for deep neural networks to be trained for hundreds of thousands of iterations. For a more insightful analysis and understanding, it is necessary to examine actual network predictions at intermediate stages of training. Raw data, ground truth, predictions, loss, and other data as requested can be summarized and written to disk in *snapshots*. Writing snapshots to disk is relatively slow compared to a network training iteration. In order to minimize GPU idle time and reduce disk usage, snapshots are created only sparsely, usually every few hundred or thousand training iterations. While it is certainly possible to use a combination of Linux command line tools to find the latest snapshot, it is more intuitive and accessible to use the `list-latest-snapshot` command that is shipped with EQIP:

```
$ list-latest-snapshot --help
usage: list-latest-snapshot [-h] [--experiment EXPERIMENT]
```



(a) a_1, a_5, a_9 (b) a_2, a_6, a_{10} (c) a_3, a_7, a_{11} (d) a_4, a_8, a_{12}

Figure 56: Visualization of snapshot of quasi-isotropic network after 30751 training iterations with the `paintera-show-container` command: Affinity prediction channels are overlaid over raw data.

```
                        [--setup SETUP [SETUP ...]]
                        [--snapshots-directory SNAPSHOTS_DIRECTORY]

optional arguments:
  -h, --help            show this help message and exit
  --experiment EXPERIMENT, -e EXPERIMENT
                        Path to experiment directory. Defaults to current
                        working directory.
  --setup SETUP [SETUP ...]
                        Print latest snapshot for these setups. Defaults to
                        all setups.
  --snapshots-directory SNAPSHOTS_DIRECTORY
                        Sub-directory of experiment that contains the
                        snapshots. Defaults to `snapshots'
```

The `paintera-show-container` command (section 4.1) can then be used to visualize the latest snapshot, *e.g.* for setup 0 of the fictious quasi-isotropic (chapter 3) experiment from section 4.2.2.1:

```
$ list-latest-snapshot \
      --experiment=$HOME/my-experiment \
      --setup=0
30751
$ paintera-show-container \
      --channels 8,4,0 9,5,1 10,6,2 11,7,3 \
      --revert-array-attributes \
      $HOME/my-experiment/0/snapshots/batch_30751.hdf \
      --exclude='.*' \
      --include '.*raw.*' '/volumes.*glia.*' '.*prediction.*'
```

The `--channels` groups affinity channels of ground truth, predictions, and loss by offset magnitude in voxels. The `--revert-array-attributes` option is necessary because array attributes like `"offset"` and `"resolution"` are stored in reversed order in the snapshot. Datasets matching any of the regular expression patterns provided with the `--exclude` option are ignored but regular expressions provided with the `--include` option can be used to override exclusion for matching datasets. Here, we add the raw data, all predictions, and glial cell ground truth annotations. The `--channels` options groups the affinity predictions into multi-channel datasets (a_1, a_5, a_9), (a_2, a_6, a_{10}), (a_3, a_7, a_{11}), and (a_4, a_8, a_{12}). figure 56 shows the visualization of training snapshot 30751 with the `paintera-show-container` command.

In a similar way, the latest checkpoint can be inquired with the `list-latest-checkpoint` command:

```
$ list-latest-checkpoint --help
usage: list-latest-checkpoint [-h] [--experiment EXPERIMENT]
                              [--setup SETUP [SETUP ...]]
                              [--checkpoint-filename CHECKPOINT_FILENAME]

optional arguments:
  -h, --help            show this help message and exit
  --experiment EXPERIMENT, -e EXPERIMENT
                        Path to experiment directory. Defaults to current
                        working directory.
  --setup SETUP [SETUP ...]
                        Print latest snapshot for these setups. Defaults to
                        all setups.
  --checkpoint-filename CHECKPOINT_FILENAME
                        File that stores latest checkpoint (relative to
                        experiment). Defaults to `checkpoint'
```

This is particularly useful for using the network for prediction at its latest training stage.

4.3 IMGLYB

The multi-dimensional `numpy.ndarray` (NumPy array) of the *NumPy*[65] (Van Der Walt, Colbert, and Gael Varoquaux 2011) Python library is the essential data structure for multi-dimensional image processing and analysis in the Python programming language. Many multi-image processing libraries use NumPy arrays as their core data structure, *e.g.* scikit-image[66] (Van der Walt et al. 2014), or expose NumPy array compatible Python interfaces, *e.g.* VIGRA,[67] ITK,[68] or OpenCV.[69] The NumPy array is a pointer into native memory with additional meta information like the size (shape) or the data type. The NumPy library and, by extension, all Python libraries with appropriate interfaces provide a wealth of efficient vectorized functions that would otherwise be inefficient if implemented in Python. Essentially, the NumPy library and ecosystem have been a major contributor to the success of the Python programming language in scientific computing, including bioimage informatics.

Java programs are executed inside a Java virtual machine (JVM) and cannot access native memory including NumPy arrays. ImageJ[70] (Schneider, Rasband, and Eliceiri 2012; Schindelin, C. T. Rueden, et al. 2015) is the most important multi-dimensional image processing and analysis frameworks for bioinformatics in Java. The Fiji[71] (Schindelin, Arganda-Carreras, et al. 2012) distribution of ImageJ ships with a large number of plugins, *e.g.* TrakEM2[72] (Cardona et al. 2012) or Trainable Weka Segmentation[73] (Arganda-Carreras et al. 2017), and contributors can provide their software through separate update sites.[74]

[65] https://www.numpy.org
[66] https://scikit-image.org
[67] https://ukoethe.github.io/vigra
[68] https://itk.org
[69] https://opencv.org
[70] https://imagej.net
[71] https://fiji.sc
[72] https://imagej.net/TrakEM2
[73] https://imagej.net/Trainable_Weka_Segmentation
[74] https://imagej.net/Update_Sites

While many scientists use both environments for their research, the lack of direct interaction results in rather crude and cumbersome solutions: Typically, part of an analysis is conducted in Python/ImageJ. The results are then written to file and loaded for further analysis in ImageJ/Python. This requires, of course, that compatible file formats are available in both environments. Instead, many scientists prefer to use frameworks like the popular Jupyter Notebooks (Kluyver et al. 2016) for execution in a single process, documentation, and publication of entire analyses, which is impossible when using both Python and ImageJ.

Modern ImageJ2 (C. T. Rueden, Schindelin, et al. 2017; C. T. Rueden and Eliceiri 2018) builds upon the ImgLib2 (Pietzsch, Preibisch, et al. 2012) Java library for generic multi-dimensional image processing. One of the core features of ImgLib2 is the separation of data representation and access of voxel values and coordinates: Virtualization of pixel access at minimal performance penalty allows to transparently exchange the underlying data representation. In other words, the caller of an ImgLib2 data structure can access the voxel values without any knowledge of how the underlying data is provided.

The `RandomAccessibleInterval`[75] interface is the ImgLib2 representation of voxel data in a discrete and finite multi-dimensional grid. Implementations of this interface can provide data through arbitrary backends or *accesses*: The `ListImg`[76] is backed by a `List`[77] of individual object instances. The `PlanarImg`[78] represents data as a list of image planes backed by primitive type arrays. The `ArrayImg`[79] holds data in a single primitive type array for efficient access. The `CellImg`[80] divides the coordinate domain into a grid of cells (blocks, chunks), each backed by a primitive type array, for image sizes that exceed the capacity of Java arrays ($\approx 2^{31}$). Other implementations do not hold their own data: The `ConvertedRandomAccessibleInterval`[81] converts pixel values as requested and avoids unnecessary copies of the data. The `StackView`[82] combines a stack of n-dimensional `RandomAccessibleInterval` into a single $n+1$-dimensional `RandomAccessibleInterval`.

This virtualization makes shared memory between ImgLib2 and NumPy data structures possible. I extended the ImgLib2 `ArrayImg` with data backends that access native memory in the *imglib2-unsafe*[83] and *imglib2-imglyb*[84] Java libraries. I developed the *imglyb* Python library that uses the Python-Java bridge *PyJNIus*[85] to expose imglib2-imglyb in Python for shared memory access between NumPy arrays and ImgLib2 in a single process. In particular, ImgLib2 can access native NumPy array memory without copying data. Conversely, NumPy can access only ImgLib2 data structures that were created with native memory backend.

---

[75] `net.imglib2.RandomAccessibleInterval`
[76] `net.imglib2.img.list.ListImg`
[77] `java.util.List`
[78] `net.imglib2.img.planar.PlanarImg`
[79] `net.imglib2.img.array.ArrayImg`
[80] `net.imglib2.img.cell.CellImg`
[81] `net.imglib2.converter.read.ConvertedRandomAccessibleInterval`
[82] `imglib2.view.StackView`
[83] https://github.com/imglib/imglib2-unsafe
[84] https://github.com/imglib/imglib2-imglyb
[85] https://github.com/kivy/pyjnius

First, I will review Python-Java bridge frameworks that follow different strategies to enable interaction of Python and the JVM.

4.3.1 *Python-Java Bridges*

Jython[86] is a JVM based re-implementation of CPython — the reference implementation in C of the Python programming language. As such, it inter-operates flawlessly with Java but the wealth of CPython extensions like NumPy cannot be used through Jython. *JyNI* (Richthofer 2014; Richthofer 2016) uses the Java Native Interface[87] (JNI) to make native CPython extensions like NumPy accessible from Jython. The general lack of support for Python 3 in Jython renders its future usefulness questionable, in particular considering the end of lifetime of Python 2 on January 1st, 2020[88] — many major projects have pledged to drop support for Python 2 by that time.[89]

Java Embededded Python[90] (Jep) — contrary to the other frameworks reviewed in this section — starts CPython interpreters from within a JVM through the JNI. Jep supports conversion between some Java and Python types, in particular Java primitive arrays and `numpy.ndarray`. Instead of sharing memory, however, data is copied during conversion.[91]

Py4J[92] starts two independent processes for Python and Java. Data and objects are shared through sockets, *i.e.* by copy and not by shared memory.

Jpy[93] is unique insofar as it is a bi-directional Python bridge: Java classes and methods can be called from Python and vice versa. Furthermore, it is possible to wrap Python classes as implementations of Java interfaces in Java — to my understanding it is not possible to pass Python classes as interfaces to Java methods from Python code. Disambiguation of overloaded methods in an interface is not possible in Python classes.

In a similar way, *JPype*[94] interfaces between Java and Python through the JNI. Java interfaces can be implemented in Python and passed as call-back to Java methods called from Python but overloaded methods cannot be disambiguated.

Finally, *PyJNIus* builds on the JNI as well: Java classes and methods can be called from Python and interfaces can be implemented with disambiguation of overloaded methods through Python decorators — the exact signature must be specified for each implemented method.

Only JNI based Python-Java bridges enable shared memory between NumPy and ImgLib2 data structures. Out of these, only PyJNIus supports implementation

[86] https://www.jython.org
[87] https://docs.oracle.com/javase/8/docs/technotes/guides/jni
[88] https://github.com/python/devguide/pull/344
[89] https://python3statement.org
[90] https://github.com/ninia/jep
[91] Java arrays are fundamentally different from c-style arrays (pointers).
[92] https://www.py4j.org/
[93] https://github.com/bcdev/jpy
[94] https://github.com/jpype-project/jpype

of arbitrary Java interfaces with overloaded methods in Python: JPype[95] and Jpy[96] match Python and Java methods by name and Python does not support method overloading. PyJNIus uses decorators to specify the exact method signature and to unambiguously map Python methods to Java methods.[97] At the time of the creation of imglyb, I was unable to implement any Java interfaces in JPype or JPy without producing segmentation faults.[98] PyJNIus was therefore the logical choice for a Python/NumPy-ImgLib2 bridge.

4.3.2 *Architecture & Design*

The ImgLib2 `ArrayImg`[99] uses *accesses* as an abstraction of the underlying data. In the standard implementation, accesses are backed by Java primitive arrays such as `long[]`, `double[]`, etc. While similar in name and function, native C-arrays — simply pointers to contiguous native memory — differ entirely from Java arrays — objects in the JVM heap that can be fragmented. It is thus impossible to use an `ArrayImg` backed by Java arrays for shared memory access of a `numpy.ndarray` that is backed by a C-array. Thanks to the abstraction of data through accesses, it is possible to implement alternative accesses that understand C-arrays — or rather a memory address: The `Unsafe`[100] class exposes native memory for allocation and read/write-access to the JVM — of course guarantees such as safe memory access are lost and segmentation faults may occur for invalid memory access. This is a problem in particular with two independent garbage collectors for Python and Java, respectively, but the solution is simple: The memory owner — typically the object that allocated the memory — must not be destroyed or garbage collected while any other object holds a reference to that memory location. I developed the imglib2-unsafe Java package to provide these accesses. Furthermore, as `ArrayImg` is limited to about 2^{31} – array size is limited by the maximum value of signed 32bit integer and loops are optimized for integer but not for long indices in Java — and a `numpy.ndarray` can allocate arbitrarily large amounts of native memory, I implemented the `UnsafeImg`[101] to support such use-cases. In practice, however, it is usually preferable to process large data in small chunks.

The NumPy-ImgLib2 bridging layer is split into the `imglib2-imglyb`[102] Java and `imglyb`[103] Python packages for Java and Python layers, respectively. These packages address, most importantly, reference management to ensure validity of memory addresses and mapping of NumPy meta data such as data types or strides into ImgLib2 space.

---

[95] https://jpype.readthedocs.io/en/latest/userguide.html#jproxy

[96] https://jpy.readthedocs.io/en/latest/intro.html#implementing-java-interfaces-using-python

[97] https://pyjnius.readthedocs.io/en/stable/api.html#jnius.java_method

[98] Re-visiting while writing this thesis, segmentation faults to not seem to be an issue anymore.

[99] `net.imglib2.img.array.ArrayImg`

[100] The internal `sun.misc.Unsafe` class is not part of the Java API and thus subject to potentially breaking changes or removal in subsequent releases.

[101] `net.imglib2.img.unsafe.UnsafeImg`

[102] https://github.com/imglib/imglib2-imglyb

[103] https://github.com/imglib/imglyb

Core functionality that is independent of image processing — in partiuclar the use of jgo (section 4.4) to resolve Apache Maven dependencies — have since been moved to the scyjava[104] package for availability and use in unrelated projects, most notably the ImageJ-Python integration pyimagej.[105]

4.3.3 *Installation*

The easiest way to install imglyb is through conda:

```
$ conda install -c conda-forge imglyb
```

This installs the PyJNIus and OpenJDK dependencies from the conda-forge channel. If a different JDK — *e.g.* Oracle JDK or OpenJDK8 with JavaFX — is preferred, PyJNIus has to be built separately[106] and the `PYJNIUS_JAR` environment variable has to be set appropriately (build instructions for Unix-like operating systems; check the Makefile and adapt as needed for installations on Windows). Apache Ant[107] is required for the build process and Apache Maven[108] at runtime.

```
$ export JAVA_HOME= # specify JDK
$ PYJNIUS_SRC_DIR=  # specify directory for PyJNIus sources
$ git clone https://github.com/kivy/pyjnius.git $PYJNIUS_SRC_DIR
$ cd $PYJNIUS_SRC_DIR
$ make
$ make tests
$ pip install $PYJNIUS_SRC_DIR
$ export PYJNIUS_JAR=$PYJNIUS_SRC_DIR/build/pyjnius.jar
```

Now, imglyb can be installed via the Python package installer for Python pip:[109]

```
$ pip install imglyb
```

The latest imglyb development version is available on GitHub.[110]

4.3.4 *Usage*

The environment variables `JAVA_HOME` and `PYJNIUS_JAR` need to be set appropriately as described in section 4.3.3. Imglyb implicitly imports PyJNIus and PyJNIus — the Python module is `jnius` — immedietaly starts a JVM when imported into Python. Thus, any Java settings and the classpath need to be specified before imglyb or PyJNIus are imported with any of these import statements:

```
import imglyb
import jnius
from jnius import ...
```

In general, imglyb should be imported before PyJNIus. The sole exception is `jnius_config`, which can be used to set JVM options and the classpath:

[104] https://github.com/scijava/scyjava
[105] https://github.com/imagej/pyimagej
[106] https://github.com/kivy/pyjnius#usage-on-desktop
[107] https://ant.apache.org
[108] https://maven.apache.org
[109] https://pip.pypa.io/en/stable
[110] https://github.com/imglib/imglyb

```
import jnius_config
jnius_config.set_classpath(*paths)
jnius_config.add_classpath(*paths)
jnius_config.set_options(*options)
jnius_config.add_options(*options)
```

In a similar way, the version of the imglib2-imglyb Maven artifact can be specified through `imglyb_config`:

```
import imglyb_config
imglyb_config.set_imglib2_imglyb_version
```

Additional Maven dependencies and jgo (section 4.4) settings can be manipulated trough `scyjava_config`:

```
import scyjava_config
# add additional maven dependencies
scyjava_config.add_endpoints(*endpoints)
# add repositories for maven dependencies
scyjava_config.add_repositories(*repos)
```

For convenience, `scyjava_config` also exposes the functions in `jnius_config`. In general, Python code that uses imglyb should be structured like this:

```
# optional:
import scyjava_config
# configure scyjava/pyjnius as needed
# optional:
import imglyb_config
# configure imglyb as needed

# import imglyb before jnius
import imglyb

# optional: import jnius as needed
import jnius
from jnius import ...

# actual code goes here
```

The most important functions in imglyb convert `numpy.ndarray` to an ImgLib2 `RandomAccessibleInterval`,[111] or — in the reverse direction — from an ImgLib2 `ArrayImg` backed by native access to a `numpy.ndarray`:

```
import imglyb
import numpy as np

# numpy.ndarray to ImgLib2:
# create some dummy data:
array = np.random.rand(10, 20, 30) * 256
img   = imglyb.to_imglib(array)
# interpret data as ARGB
# numpy.uint32 and numpy.int32 only:
argb_arr = array.astype(np.int32)
argb_img = imglyb.to_imglib_argb(argb_arr)

# ImgLib2 to numpy.ndarray:
# create some data, here we just
# make a copy of the converted img
from jnius import autoclass
package    = 'net.imglib2.python'
clazz      = 'ArrayImgWithUnsafeStoreFactory'
class_name = '%s.%s' % (package, clazz)
```

[111] `net.imglib2.python.ReferenceGuardingRandomAccessibleInterval`

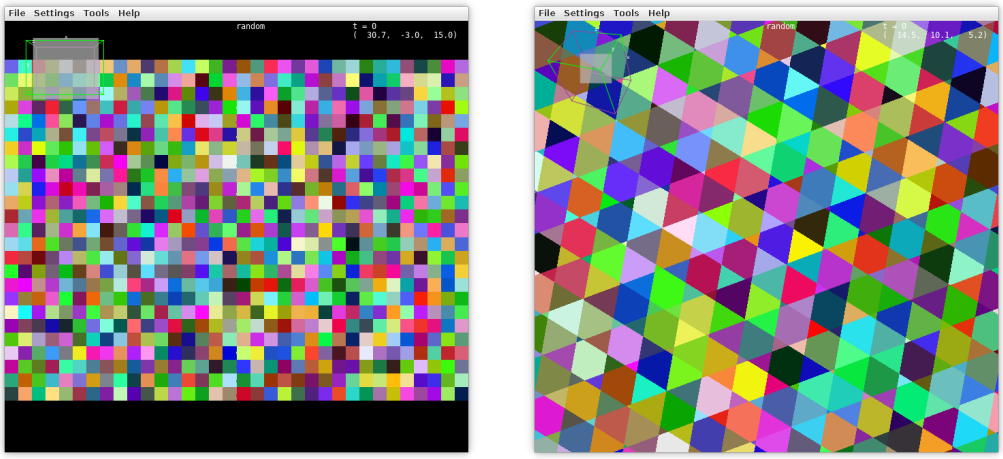
```
Factory    = autoclass(class_name)
factory    = Factory(img.randomAccess().get())
copy       = factory.create(img)

# copy (slow, do not use for large images)
s = imglyb.util.Views.flatIterable(img).cursor()
t= imglyb.util.Views.flatIterable(copy).cursor()
while s.hasNext():
    t.next().set(s.next())
from_img = imglyb.to_numpy(copy)
# converted copy should be element-wise
# equal to original
assert np.all(from_img == array)
```

Additionally, visualization of data through `BigDataViewer` is made available in the `imglyb.util` module. Python is unaware of any JVM threads and the Python interpreter — unless in interactive mode — needs to be kept alive as desired:

```
import imglyb
import imglyb.util as util
import numpy as np

# create random ARGB data
data = np.random.randint(
    0,
    2**32,
    size=(30, 25, 35),
    dtype=np.uint32)
img = imglyb.to_imglib_argb(data)
bdv = util.BdvFunctions.show(img, 'random')

# python does not know about Java threads
# instead of keeping python interpreter alive
# infinitely, keep it alive only while window is showing
from jnius import autoclass
import threading
import time
package = 'net.imglib2.python'
clazz   = 'BdvWindowClosedCheck'
check   = autoclass('%s.%s' % (package, clazz))()
swing   = autoclass('javax.swing.SwingUtilities')
vp      = bdv.getBdvHandle().getViewerPanel()
frame   = swing.getWindowAncestor(vp)
frame.addWindowListener(check)

def sleep():
    while check.isOpen():
        time.sleep(1.0)

t = threading.Thread(target=sleep)
t.start()
t.join()
```

Figure 57 shows the BigDataViewer window as created by above Python code. Unfortunately, due to an incompatibility between the Java awt and Cocoa event loops, Java awt applications cannot currently be started from Python on OSX. Donald Olbris contributed a wrapper script that ensures that the respective event loops are started in the correct order and Java awt applications can be used on OSX as well.[112]

[112] https://github.com/imglib/imglyb#awt-through-pyjnius-on-osx



(a) (b)

Figure 57: Visualization of a random integer-valued NumPy array in BigDataViewer through imglyb; the values are interpreted as ARGB color: Initial state of the viewer (a) , and after arbitrary navigation (b). Cross-sections can be arbitrarily oriented. Rendering and navigation are fast and responsive.

Visualization is not restricted to BigDataViewer: In a more comprehensive effort, I developed the payntera[113] package to expose Python bindings for Paintera. Meshes can be generated on the fly from a `numpy.ndarray` and rendered inside the Paintera 3D-view. In contrast to Java awt, no wrapper script is required to start JavaFX applications in Python on OSX. JavaFX requires explicit control of the event loop, which is also handled in the payntera package:[114]

```
# set heap size to reasonable value
import jnius_config
jnius_config.add_options('-Xmx2g')

import numpy as np
import payntera
import payntera.jfx
import scipy.ndimage
import time

# imglyb and jnius must be imported after
# payntera is imported!
import imglyb
# jnius must be imported after imglyb is imported!
from jnius import autoclass, JavaException

payntera.jfx.init_platform()

package          = 'org.janelia.saalfeldlab.paintera'
clazz            = 'PainteraBaseView'
class_name       = '%s.%s' % (package, clazz)
PainteraBaseView = autoclass(class_name)
viewer           = PainteraBaseView.defaultView()
pbv              = viewer.baseView
pane             = viewer.paneWithStatus.getPane()
scene, stage     = payntera.jfx.start_stage(pane)

```

[113] https://github.com/saalfeldlab/payntera

[114] https://github.com/saalfeldlab/payntera/blob/3f28f130/example-blobs.py

```
# generate some data
shape      = (80,80,50)
x, y, z    = np.indices(shape)
fx, fy, fz = 2              \
  * np.pi / np.array(shape) \
  * np.array([10, 1, 3])

raw = (1+np.sin(x * fx))          \
  * (1+np.sin(y * fy))            \
  * (1+x*y/(shape[0]*shape[1]))**2 \
  * (1+np.cos(z * fz))            \
  * ((x+y+z)/np.sum(shape))
raw_img    = imglyb.to_imglib(raw)
labels, nb = scipy.ndimage.label(raw > 0.5)
labels_img = imglyb.to_imglib(labels)

# show in Paintera
raw_state = pbv.addSingleScaleRawSource(
    raw_img,
    [1.0, 1.0, 1.0], # resolution
    [0.0, 0.0, 0.0], # offset
    np.min(raw),     # contrast min
    7,               # contrast max
    'raw')           # name
state     = pbv.addSingleScaleLabelSource(
    labels_img,
    [1.0, 1.0, 1.0], # resolution
    [0.0, 0.0, 0.0], # offset
    nb+1,            # max label
    'labels')        # name

viewer.keyTracker.installInto(scene)
ANY = autoclass('javafx.scene.input.MouseEvent').ANY
scene.addEventFilter(ANY, viewer.mouseTracker)

while stage.isShowing():
    time.sleep(1.0)
```

The synthetic data from this example is visualized in figure 58. Payntera has been particularly useful for the visual inspection of data augmentations for the training (section 4.2) of a quasi-isotropic neural network architecture (section 3.4) that were fully implemented in Python. More advanced examples — *e.g.* the implementation of a brush in Python to paint into a `numpy.ndarray` with BigDataViewer or image processing with NumPy and ImgLib2 and shared memory — and tutorials in the form of Jupyter notebooks are available on GitHub.[115,116]

4.3.5 *Discussion & Future Work*

With imglyb, I was able to create a bridge between two of the most widely used frameworks in (biological) multi-dimensional image processing and, hopefully, researchers will be able to benefit from combining both frameworks in analysis pipelines with shared memory access. While, at the moment, imglyb is used mostly by the ImageJ community, I will present imglyb at the SciPy conference 2019 (Hanslovsky 2019) and, with increased awareness, hopefully, some of the remaining issues can be addressed by the open-source Python community: Current

[115] https://github.com/hanslovsky/imglyb-learnathon
[116] https://github.com/hanslovsky/imglyb-examples

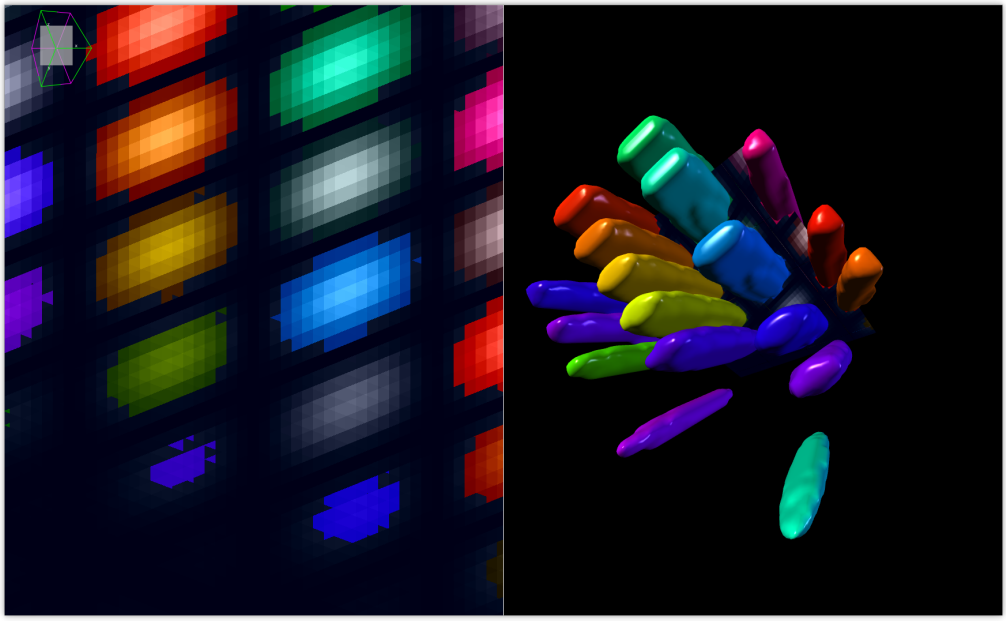


Figure 58: Paintera visualizes NumPy arrays with on-the-fly mesh generation and 3D rendering for label data. The raw data is synthesized as a product of simple trigonometric functions and polynomials. The labels are generated as connected components on the thresholded raw data.

lead developers and maintainers are proficient mostly in JVM-based languages and some of the issues below may have causes in low-level Python functionality, *e.g.* garbage collection and reference counting.

Currently, use of imglyb is restricted to relatively simple scenarios: NumPy arrays need to remain in scope when being wrapped into ImgLib2 data structures. This is most likely caused by a reference counting issue inside PyJNIus.[117] Attempts to visualize multi-dimensional chunked arrays like dask arrays (Rocklin 2015) in BigDataViewer or Paintera have so far been unsuccessful. I suspect that this is related to the same reference counting issue. Currently, `pyjnius.jar` is required to be available locally as specified by the `PYJNIUS_JAR` enviornment variable even though `pyjnius.jar` is not platform-dependent. This could be simplified by distributing `pyjnius.jar` through a Java package manager, *e.g.* Apache Maven, and downloading `pyjnius.jar` as needed. On top of that, auto-detection of the JDK when `JAVA_HOME` is not set is currently discussed for the upstream `scyjava` library.[118]

4.4 JGO

Apache Maven[119] is a dependency manager and build automation tool for Java. The central element of a maven project is the project object model (POM) that specifies a so-called *artifact*. A Maven artifact is identified by a *coordinate* with

[117] https://github.com/kivy/pyjnius/issues/345
[118] https://github.com/scijava/scyjava/pull/7
[119] https://maven.apache.org

three components: `groupId` — the owner or context of the package, `artifactId` — the name of the package, and its `version`. Typically, a coordinate is specified as a colon-separated string:

```
groupId:artifactId:version
```

Artifacts can be distributed and made available through Maven repositories such as the Central Repository,[120] or the ImageJ Maven repository.[121] Artifacts that are installed or downloaded through maven are cached locally. While it is possible to compile and execute an artifact from a POM file

```
$ # brackets indicate optional arguments
$ mvn -Pexec [-f /path/to/pom.xml]
```

an artifact cannot be executed just from its coordinates, even if it is available through a repository. To address this, Curtis Rueden wrote *jgo*,[122] formerly known as *jrun*, a bash script that downloads and executes a maven artifact identified by its coordinates and its dependencies from online repositories. A local cache at `~/.jgo` speeds up subsequent executions of the same artifact. By default, jar files are hard-linked from the local maven repository to minimize the storage footprint if possible. I realized that jgo would be a great way to make Java applications — *e.g.* Paintera — available for non-expert users that are not familiar with Maven or the general software build and compilation process. While the bash language is available on Unix(-like) operating systems like macOS or Linux, distributing to Windows operating systems would be challenging. Instead, I decided to re-implement jgo in an operating system agnostic way using the Python programming language. Users can now simply install jgo through the package installer for Python pip:[123]

```
$ pip install jgo
```

or conda

```
$ conda install -c conda-forge jgo
```

As a result, JVM-based applications can be installed simply through pip or conda; the only additional non-Python dependencies are Java and Maven. The latest development version is available on the jgo GitHub repository.

4.4.1 *Usage*

Jgo tries to simplify execution of Maven artifacts as much as possible and thus employs a simple synopsis:

```
$ # brackets indicate optional arguments
$ jgo [jgo and java options] endpoint [program options]
```

The endpoint consists of one or more Maven coordinates, separated by a `+'-sign. In more detail (additional line-breaks introduced for better readabilty):

[120] https://search.maven.org
[121] https://maven.imagej.net
[122] https://github.com/scijava/jgo
[123] https://pip.pypa.io

```
$ jgo --help
usage: jgo [-v] [-u] [-U] [-m] [--ignore-jgorc] [--link-type type] [--additional-jars
           jar [jar ...]] [--additional-endpoints endpoint [endpoint ...]]
           [JVM_OPTIONS [JVM_OPTIONS ...]] <endpoint> [main-args]

Run Java main class from maven coordinates.

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         verbose mode flag
  -u, --update-cache    update/regenerate cached environment
  -U, --force-update    force update from remote Maven repositories (implies -u)
  -m, --manage-dependencies
                        use endpoints for dependency management (see "Details" below)
  -r REPOSITORY [REPOSITORY ...], --repository REPOSITORY [REPOSITORY ...]
                        Add additional maven repository (key=url format)
  -a ADDITIONAL_JARS [ADDITIONAL_JARS ...], --additional-jars ADDITIONAL_JARS
                                                              [ADDITIONAL_JARS ...]
                        Add additional jars to classpath
  --additional-endpoints ADDITIONAL_ENDPOINTS [ADDITIONAL_ENDPOINTS ...]
                        Add additional endpoints
  --ignore-jgorc        Ignore ~/.jgorc
  --link-type {hard,soft,copy,auto}
                        How to link from local maven repository into jgo cache.
                        Defaults to the `links' setting in ~/.jrunrc or 'auto'
                        if not specified.

The endpoint should have one of the following formats:

- groupId:artifactId
- groupId:artifactId:version
- groupId:artifactId:mainClass
- groupId:artifactId:version:mainClass
- groupId:artifactId:version:classifier:mainClass

If version is omitted, then RELEASE is used.
If mainClass is omitted, it is auto-detected.
You can also write part of a class beginning with an @ sign,
and it will be auto-completed.
```

If no main class is specified in the Maven artifact or to run a different main class, either of the `groupId:artifactId:mainClass` or `groupId:artifactId:version:mainClass` endpoint syntaxes is used. The `mainClass` of an additional endpoint is ignored if specified. Part of the fully qualified `mainClass` — starting at the beginning of the class name — can be replaced by a single wildcard `@`-sign to avoid overly verbose commands.

4.4.1.1 *Configuration*

Jgo can be configured with an optional configuration file at `~/.jgroc` that is structured similarly to a Windows INI file:[124] properties in three sections — `repositories`, `settings`, and `shortcuts` — are specified as `='`-separated key-value pairs, *e.g.*.

```
[repositories]
; add ImageJ repository to resolve Maven artifacts
imagej = https://maven.imagej.net/content/groups/public
[settings]
; location of the local Maven repository
; defaults to ~/.m2/repository if not specified
```

[124] https://en.wikipedia.org/wiki/INI_file

```
; replace with valid path:
m2Repo = /path/to/.m2Repo
; location of the jgo cache dir
; defaults to ~/.jgo if not specified
; replace with valid path:
cacheDir = /path/to/.jgo
; change link type for jars in jgo cache:
;   - hard
;   - soft
;   - copy
;   - auto (default) uses hard links if possible,
;                    soft links otherwise
links = soft
[shortcuts]
repl   = imagej:org.scijava.script.ScriptREPL
imagej = net.imagej:imagej
```

Shortcuts are expanded per endpoint from the beginning of the string until no further expansion is possible. With this config file, for example, the invocation of the SciJava script REPL[125]

```
$ jgo \
    -r ij=https://maven.imagej.net/content/groups/public \
    net.imagej:imagej:org.scijava.script.ScriptREPL
```

is simplified to just

```
$ jgo repl
```

In the following, I will present a few usage examples to help explain and build intuition for the synopsis.

4.4.1.2 *Examples*

The command line interface of the Parsington[126]— the SciJava[127] parser for mathematical expressions— can simply be invoked without the need for installation:

```
$ jgo org.scijava:parsington 1+3
4
```

Omitting the command line argument will start an interactive console shell. Additional jars can be added through the `+'-notation or the `--additional-endpoints` option. This is useful to modify behavior or add functionality ao a program. For example, the SciJava script REPL interprets different programming languages depending on the classpath (assuming repositories to be configured as in section 4.4.1.1):

```
# JRuby:
$ jgo org.scijava:scijava-common:@REPL+org.scijava:scripting-jruby
$ jgo \
    --additional-endpoints org.scijava:scripting-jruby \
    org.scijava:scijava-common:@REPL
# Jython:
$ jgo org.scijava:scijava-common:@REPL+org.scijava:scripting-jython
$ jgo \
    --additional-endpoints org.scijava:scripting-jython \
    org.scijava:scijava-common:@REPL
# Groovy:
```

[125] https://github.com/scijava/scijava-common
[126] https://github.com/scijava/parsington
[127] https://scijava.org

```
$ jgo org.scijava:scijava-common:@REPL+org.scijava:scripting-groovy
$ jgo \
    --additional-endpoints org.scijava:scripting-groovy \
    org.scijava:scijava-common:@REPL
```

Similarly, logging backends for the popular Java logging facade framework SLF4J[128] can be easily exchanged as preferred (assuming repositories to be configured as in section 4.4.1.1):

```
# No logging:
$ jgo org.janelia.saalfeldlab:paintera
# slf4j-simple:
$ jgo org.janelia.saalfeldlab:paintera+org.slf4j:slf4j-simple:1.7.25
# slf4j-log4j12:
$ jgo org.janelia.saalfeldlab:paintera+org.slf4j:slf4j-log4j12:1.7.25
# slf4j-jdk14:
$ jgo org.janelia.saalfeldlab:paintera+org.slf4j:slf4j-jdk14:1.7.25
# logback:
$ jgo org.janelia.saalfeldlab:paintera+ch.qos.logback:logback-classic:1.2.3
```

4.4.2 *Discussion*

By channeling the powerful dependency management of Apache Maven, jgo makes running Maven-based JVM applications straightforward and convenient. The Python implementation makes distribution of applications extremely easy. Python jgo can also be used to populate the classpath from Maven dependencies when using Java-Python bridging frameworks (section 4.3).

[128] https://www.slf4j.org

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. "Tensorflow: A system for large-scale machine learning". In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[2] Ernst Abbe. "Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung". In: *Archiv für mikroskopische Anatomie* 9.1 (1873), pp. 413–418.

[3] Bjoern Andres, Thorsten Beier, and Jörg H Kappes. "OpenGM: A C++ library for discrete graphical models". In: *arXiv preprint arXiv:1206.0111* (2012).

[4] Bjoern Andres, Jörg H Kappes, Thorsten Beier, Ullrich Köthe, and Fred A Hamprecht. "Probabilistic image segmentation with closedness constraints". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 2611–2618.

[5] Björn Andres, Ullrich Köthe, Moritz Helmstaedter, Winfried Denk, and Fred A Hamprecht. "Segmentation of SBFSEM volume data of neural tissue by hierarchical classification". In: *Joint Pattern Recognition Symposium*. Springer. 2008, pp. 142–152.

[6] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2011.

[7] David Applegate, Ribert Bixby, Vasek Chvatal, and William Cook. *Concorde TSP solver*. 2006.

[8] Ignacio Arganda-Carreras, Verena Kaynig, Curtis Rueden, Kevin W Eliceiri, Johannes Schindelin, Albert Cardona, and H Sebastian Seung. "Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification". In: *Bioinformatics* 33.15 (2017), pp. 2424–2426.

[9] Frank J Ayd. *Lexicon of psychiatry, neurology, and the neurosciences*. Lippincott Williams & Wilkins, 2000.

[10] Shai Bagon and Meirav Galun. "Large scale correlation clustering optimization". In: *arXiv preprint arXiv:1112.2903* (2011).

[11] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. "Correlation clustering". In: *Machine learning* 56.1-3 (2004), pp. 89–113.

[12] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[13] Richard Beare and Gaëtan Lehmann. "The watershed transform in ITK-discussion and new developments". In: *The Insight Journal* 6 (2006).

[14] Thorsten Beier. "Multicut Algorithms for Neurite Segmentation". PhD thesis. 2018.

[15] Thorsten Beier, Constantin Pape, Nasim Rahaman, Timo Prange, Stuart Berg, Davi D Bock, Albert Cardona, Graham W Knott, Stephen M Plaza, Louis K Scheffer, et al. "Multicut brings automated neurite segmentation closer to human performance". In: *Nature Methods* 14.2 (2017), p. 101.

[16] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. "Theano: a CPU and GPU math expression compiler". In: *Proceedings of the Python for scientific computing conference (SciPy)*. Vol. 4. 3. Austin, TX. 2010.

[17] M. L. Berlanga, S. Phan, E. A. Bushong, S. Wu, et al. "Three-Dimensional Reconstruction of Serial Mouse Brain Sections: Solution for Flattening High-Resolution Large-Scale Mosaics". In: *Frontiers in Neuroanatomy* 5 (Mar. 2011). ISSN: 1662-5129. DOI: 10.3389/fnana.2011.00017.

[18] Davi D. Bock, Wei-Chung Allen Lee, Aaron M. Kerlin, Mark L. Andermann, Greg Hood, Arthur W. Wetzel, Sergey Yurgenson, Edward R. Soucy, Hyon Suk Kim, and R. Clay Reid. "Network anatomy and in vivo physiology of visual cortical neurons". In: *Nature* 471.7337 (2011), pp. 177–182.

[19] K. M. Boergens and W. Denk. "Controlling FIB-SBEM slice thickness by monitoring the transmitted ion beam". en. In: *Journal of Microscopy* 252.3 (Dec. 2013), pp. 258–262. ISSN: 1365-2818. DOI: 10.1111/jmi.12086.

[20] Kevin M Boergens, Manuel Berning, Tom Bocklisch, Dominic Bräunlein, Florian Drawitsch, Johannes Frohnhofen, Tom Herold, Philipp Otto, Norman Rzepka, Thomas Werkmeister, et al. "webKnossos: efficient online 3D data annotation for connectomics". In: *nature methods* 14.7 (2017), p. 691.

[21] Gustav Jacob Born. "Die Plattenmodellirmethode". In: *Archiv für Mikroskopische Anatomie* 22.1 (1883), pp. 584–599.

[22] Edward S Boyden, Feng Zhang, Ernst Bamberg, Georg Nagel, and Karl Deisseroth. "Millisecond-timescale, genetically targeted optical control of neural activity". In: *Nature neuroscience* 8.9 (2005), p. 1263.

[23] Yuri Boykov and Vladimir Kolmogorov. "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 9 (2004), pp. 1124–1137.

[24] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[25] K. L. Briggman and D. D. Bock. "Volume electron microscopy for neuronal circuit reconstruction". In: *Current Opinion in Neurobiology*. Neurotechnology 22.1 (Feb. 2012), pp. 154–161. ISSN: 0959-4388. DOI: 10.1016/j.conb.2011.10.022.

[26] Kevin L Briggman and Winfried Denk. "Towards neural circuit reconstruction with volume electron microscopy techniques". In: *Current opinion in neurobiology* 16.5 (2006), pp. 562–570.

[27] GY Buzsaki, Reginald G Bickford, Greg Ponomareff, LJ Thal, Ronald Mandel, and Fred H Gage. "Nucleus basalis and thalamic control of neocortical activity in the freely moving rat". In: *Journal of neuroscience* 8.11 (1988), pp. 4007–4026.

[28] SRY Cajal. "Textura del Sistema nervioso del hombre y los vertebrados. Histology of the nervous system of man and vertebrates". In: *New England Journal of Medicine* (1899). DOI: 10.1056/NEJM199510193331619.

[29] Albert Cardona, Stephan Saalfeld, Johannes Schindelin, Ignacio Arganda-Carreras, Stephan Preibisch, Mark Longair, Pavel Tomančák, Volker Hartenstein, and Rodney J Douglas. "TrakEM2 software for neural circuit reconstruction". In: *PloS one* 7.6 (2012), e38011.

[30] Rich Caruana. "Multitask learning". In: *Machine learning* 28.1 (1997), pp. 41–75.

[31] Fei Chen, Paul W Tillberg, and Edward S Boyden. "Expansion microscopy". In: *Science* 347.6221 (2015), pp. 543–548.

[32] Tsai-Wen Chen, Trevor J Wardill, Yi Sun, Stefan R Pulver, Sabine L Renninger, Amy Baohan, Eric R Schreiter, Rex A Kerr, Michael B Orger, Vivek Jayaraman, et al. "Ultrasensitive fluorescent proteins for imaging neuronal activity". In: *Nature* 499.7458 (2013), p. 295.

[33] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. "3D U-Net: learning dense volumetric segmentation from sparse annotation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2016, pp. 424–432.

[34] D. Cireșan, U. Meier, and J. Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. June 2012, pp. 3642–3649. DOI: 10.1109/CVPR.2012.6248110.

[35] Dan Cireșan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. "Deep neural networks segment neuronal membranes in electron microscopy images". In: *Advances in neural information processing systems*. 2012, pp. 2843–2851.

[36] Bert De Brabandere, Davy Neven, and Luc Van Gool. "Semantic instance segmentation with a discriminative loss function". In: *arXiv preprint arXiv: 1708.02551* (2017).

[37] D. M. De Groot. "Comparison of methods for the estimation of the thickness of ultrathin tissue sections". eng. In: *Journal of Microscopy* 151.Pt 1 (July 1988), pp. 23–42. ISSN: 0022-2720.

[38] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. "Discriminative unsupervised feature learning with convolutional neural networks". In: *Advances in neural information processing systems*. 2014, pp. 766–774.

[39] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". In: *arXiv preprint arXiv:1603.07285* (2016).

[40] Anna Lena Eberle and Dirk Zeidler. "Multi-beam scanning electron microscopy for high-throughput imaging in connectomics research". In: *Frontiers in neuroanatomy* 12 (2018), p. 112.

[41] Katharina Eichler, Feng Li, Ashok Litwin-Kumar, Youngser Park, Ingrid Andrade, Casey M Schneider-Mizell, Timo Saumweber, Annina Huser, Claire Eschbach, Bertram Gerber, et al. "The complete connectome of a learning and memory centre in an insect brain". In: *Nature* 548.7666 (2017), p. 175.

[42] Ahmed Fakhry, Hanchuan Peng, and Shuiwang Ji. "Deep models for brain EM image segmentation: novel insights and improved performance". In: *Bioinformatics* 32.15 (2016), pp. 2352–2358.

[43] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy. "Semantic instance segmentation via deep metric learning". In: *arXiv preprint arXiv:1703.10277* (2017).

[44] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Distance transforms of sampled functions". In: *Theory of computing* 8.1 (2012), pp. 415–428.

[45] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692.

[46] Jan Funke, Bjoern Andres, Fred A Hamprecht, Albert Cardona, and Matthew Cook. "Efficient automatic 3D-reconstruction of branching neurons from EM data". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1004–1011.

[47] Jan Funke, Fabian David Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. "A deep structured learning approach towards automating connectome reconstruction from 3D electron micrographs". In: *arXiv preprint arXiv:1709.02974* (2017).

[48] Jan Funke, Fabian David Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. "Large scale image segmentation with structured loss based deep learning for connectome reconstruction". In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

[49] Daniel Haehn, Seymour Knowles-Barley, Mike Roberts, Johanna Beyer, Narayanan Kasthuri, Jeff W Lichtman, and Hanspeter Pfister. "Design and evaluation of interactive proofreading tools for connectomics". In: *IEEE transactions on visualization and computer graphics* 20.12 (2014), pp. 2466–2475.

[50] Philipp Hanslovsky. "imglyb — Bridging The Chasm Between ImageJ and NumPy". In: *SciPy Conference*. 2019.

[51] Philipp Hanslovsky, John A Bogovic, and Stephan Saalfeld. "Image-based correction of continuous and discontinuous non-planar axial distortion in serial section microscopy". In: *Bioinformatics* 33.9 (2017), pp. 1379–1386.

[52] Philipp Hanslovsky, John A Bogovic, and Stephan Saalfeld. "Post-acquisition image based compensation for thickness variation in microscopy section series". In: *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*. IEEE. 2015, pp. 507–511.

[53] K. J. Hayworth, N. Kasthuri, R. Schalek, and J. W. Lichtman. "Automating the Collection of Ultrathin Serial Sections for Large Volume TEM Reconstructions". In: *Microscopy and Microanalysis* 12.Suppl. 02 (2006), pp. 86–87.

[54] Kenneth J. Hayworth, C. Shan Xu, Zhiyuan Lu, Graham W. Knott, Richard D. Fetter, Juan Carlos Tapia, Jeff W. Lichtman, and Harald F. Hess. "Ultrastructurally smooth thick partitioning and volume stitching for large-scale connectomics". In: *Nature Methods* 12.4 (2015), pp. 319–322.

[55] Larissa Heinrich, John A Bogovic, and Stephan Saalfeld. "Deep learning for isotropic super-resolution from non-isotropic 3D electron microscopy". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2017, pp. 135–143.

[56] Larissa Heinrich, Jan Funke, Constantin Pape, Juan Nunez-Iglesias, and Stephan Saalfeld. "Synaptic cleft segmentation in non-isotropic volume electron microscopy of the complete drosophila brain". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 317–325.

[57] Moritz Helmstaedter. "Cellular-resolution connectomics: challenges of dense neural circuit reconstruction". In: *Nature methods* 10.6 (2013), p. 501.

[58] Jürgen A. W. Heymann, Mike Hayles, Ingo Gestmann, Lucille A. Giannuzzi, Ben Lich, and Sriram Subramaniama. "Site-specific 3D imaging of cells and tissues with a dual beam microscope". In: *Journal of Structural Biology* 155.1 (2006), pp. 63–73.

[59] Jan Huisken, Jim Swoger, Filippo Del Bene, Joachim Wittbrodt, and Ernst HK Stelzer. "Optical sectioning deep inside live embryos by selective plane illumination microscopy". In: *Science* 305.5686 (2004), pp. 1007–1009.

[60] Viren Jain, Joseph F Murray, Fabian Roth, Srinivas Turaga, Valentin Zhigulin, Kevin L Briggman, Moritz N Helmstaedter, Winfried Denk, and H Sebastian Seung. "Supervised learning of image restoration with convolutional networks". In: *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007, pp. 1–8.

[61] Michał Januszewski, Jeremy Maitin-Shepard, Peter Li, Jörgen Kornfeld, Winfried Denk, and Viren Jain. "Flood-filling networks". In: *arXiv preprint arXiv:1611.00421* (2016).

[62] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional architecture for fast feature embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 675–678.

[63] Cory Jones, Ting Liu, Nathaniel Wood Cohan, Mark Ellisman, and Tolga Tasdizen. "Efficient semi-automatic 3D segmentation for neuron tracing in electron microscopy images". In: *Journal of neuroscience methods* 246 (2015), pp. 13–21.

[64] H. G. Jones, K. P. Mingard, and D. C. Cox. "Investigation of slice thickness and shape milled by a focused ion beam for three-dimensional reconstruction of microstructures". In: *Ultramicroscopy* 139 (Apr. 2014), pp. 20–28. ISSN: 0304-3991. DOI: 10.1016/j.ultramic.2014.01.003.

[65] William T Katz and Stephen M Plaza. "DVID: Distributed Versioned Image-Oriented Dataservice". In: *Frontiers in neural circuits* 13 (2019).

[66] Verena Kaynig, Amelio Vazquez-Reina, Seymour Knowles-Barley, Mike Roberts, Thouis R Jones, Narayanan Kasthuri, Eric Miller, Jeff Lichtman, and Hanspeter Pfister. "Large-scale automatic reconstruction of neuronal processes from electron microscopy images". In: *Medical image analysis* 22.1 (2015), pp. 77–88.

[67] Sungwoong Kim, Sebastian Nowozin, Pushmeet Kohli, and Chang D Yoo. "Higher-order correlation clustering for image segmentation". In: *Advances in neural information processing systems*. 2011, pp. 1530–1538.

[68] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[69] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Jason Grout, Sylvain Corlay, et al. "Jupyter Notebooks-a publishing format for reproducible computational workflows." In: *ELPUB*. 2016, pp. 87–90.

[70] Graham Knott, Herschel Marchman, David Wall, and Ben Lich. "Serial Section Scanning Electron Microscopy of Adult Brain Tissue Using Focused Ion Beam Milling". In: *The Journal of Neuroscience* 28.12 (2008), pp. 2959–2964.

[71] Graham Knott, Herschel Marchman, David Wall, and Ben Lich. "Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling". In: *Journal of Neuroscience* 28.12 (2008), pp. 2959–2964.

[72] Ullrich Köthe et al. "Vigra-vision with generic algorithms". In: *Cognitive Systems Group, University of Hamburg, Germany* (2008).

[73] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[74] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[75] Jeff W. Lichtman, Hanspeter Pfister, and Nir Shavit. "The big data challenges of connectomics". In: *Nature Neuroscience* 17.11 (2014), pp. 1448–1454.

[76] Jeff W Lichtman and Winfried Denk. "The big and the small: challenges of imaging the brain's circuits". In: *Science* 334.6056 (2011), pp. 618–623.

[77] Jeff W Lichtman and Joshua R Sanes. "Ome sweet ome: what can the genome tell us about the connectome?" In: *Current opinion in neurobiology* 18.3 (2008), pp. 346–353.

[78] Dayu Lin, Maureen P Boyle, Piotr Dollar, Hyosang Lee, ES Lein, Pietro Perona, and David J Anderson. "Functional identification of an aggression locus in the mouse hypothalamus". In: *Nature* 470.7333 (2011), p. 221.

[79] Ting Liu, Cory Jones, Mojtaba Seyedhosseini, and Tolga Tasdizen. "A modular hierarchical approach to 3D electron microscopy image segmentation". In: *Journal of neuroscience methods* 226 (2014), pp. 88–102.

[80] Xu Liu, Steve Ramirez, Petti T Pang, Corey B Puryear, Arvind Govindarajan, Karl Deisseroth, and Susumu Tonegawa. "Optogenetic stimulation of a hippocampal engram activates fear memory recall". In: *Nature* 484.7394 (2012), p. 381.

[81] William E Lorensen and Harvey E Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics*. Vol. 21. 4. ACM. 1987, pp. 163–169.

[82] David G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[83] Kyle Luther and H Sebastian Seung. "Learning Metric Graphs for Neuron Segmentation In Electron Microscopy Images". In: *arXiv preprint arXiv: 1902.00100* (2019).

[84] Dirk Merkel. "Docker: lightweight linux containers for consistent development and deployment". In: *Linux Journal* 2014.239 (2014), p. 2.

[85] Joshua L Morgan and Jeff W Lichtman. "Why not connectomics?" In: *Nature methods* 10.6 (2013), p. 494.

[86] Juan Nunez-Iglesias, Ryan Kennedy, Toufiq Parag, Jianbo Shi, and Dmitri B Chklovskii. "Machine learning of hierarchical clustering to segment 2D and 3D images". In: *PloS one* 8.8 (2013), e71715.

[87] Juan Nunez-Iglesias, Ryan Kennedy, Stephen M Plaza, Anirban Chakraborty, and William T Katz. "Graph-based active learning of agglomeration (GALA): a Python library to segment 2D and 3D neuroimages". In: *Frontiers in neuroinformatics* 8 (2014), p. 34.

[88] Toufiq Parag, Fabian Tschopp, William Grisaitis, Srinivas C Turaga, Xuewen Zhang, Brian Matejek, Lee Kamentsky, Jeff W Lichtman, and Hanspeter Pfister. "Anisotropic EM segmentation by 3D affinity learning and agglomeration". In: *arXiv preprint arXiv:1707.08935* (2017).

[89] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[90] Tobias Pietzsch, Stephan Preibisch, Pavel Tomančák, and Stephan Saalfeld. "ImgLib2—generic image processing in Java". In: *Bioinformatics* 28.22 (2012), pp. 3009–3011.

[91] Tobias Pietzsch, Stephan Saalfeld, Stephan Preibisch, and Pavel Tomančák. "BigDataViewer: visualization and processing for large image data sets". In: *Nature methods* 12.6 (2015), p. 481.

[92] Peter G Pitrone, Johannes Schindelin, Luke Stuyvenberg, Stephan Preibisch, Michael Weber, Kevin W Eliceiri, Jan Huisken, and Pavel Tomančák. "OpenSPIM: an open-access light-sheet microscopy platform". In: *nature methods* 10.7 (2013), p. 598.

[93] Stephen M. Plaza, Louis K. Scheffer, and Dmitri B. Chklovskii. "Toward large-scale connectome reconstructions". In: *Current Opinion in Neurobiology* 25 (2014), pp. 201–210.

[94] Steve Ramirez, Xu Liu, Pei-Ann Lin, Junghyup Suh, Michele Pignatelli, Roger L Redondo, Tomás J Ryan, and Susumu Tonegawa. "Creating a false memory in the hippocampus". In: *Science* 341.6144 (2013), pp. 387–391.

[95] William M Rand. "Objective criteria for the evaluation of clustering methods". In: *Journal of the American Statistical association* 66.336 (1971), pp. 846–850.

[96] Stefan Richthofer. "Garbage Collection in JyNI-How to bridge Mark/Sweep and Reference Counting GC". In: *arXiv preprint arXiv:1607.00825* (2016).

[97] Stefan Richthofer. "JyNI-using native CPython-extensions in Jython". In: *arXiv preprint arXiv:1404.6390* (2014).

[98] Matthew Rocklin. "Dask: Parallel computation with blocked algorithms and task scheduling". In: *Proceedings of the 14th Python in Science Conference*. 130-136. Citeseer. 2015.

[99] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[100] Curtis T Rueden and Kevin W Eliceiri. "The ImageJ ecosystem: an open and extensible platform for biomedical image analysis". In: *Microscopy Histopathology and Analytics*. Optical Society of America. 2018, MTh2A–3.

[101] Curtis T Rueden, Johannes Schindelin, Mark C Hiner, Barry E DeZonia, Alison E Walter, Ellen T Arena, and Kevin W Eliceiri. "ImageJ2: ImageJ for the next generation of scientific image data". In: *BMC bioinformatics* 18.1 (2017), p. 529.

[102] Stephan Saalfeld, Albert Cardona, Volker Hartenstein, and Pavel Tomančák. "As-rigid-as-possible mosaicking and serial section registration of large ssTEM datasets". In: *Bioinformatics* 26.12 (2010), pp. i57–i63.

[103] Stephan Saalfeld, Albert Cardona, Volker Hartenstein, and Pavel Tomančák. "CATMAID: collaborative annotation toolkit for massive amounts of image data". In: *Bioinformatics* 25.15 (2009), pp. 1984–1986.

[104] Stephan Saalfeld, Richard Fetter, Albert Cardona, and Pavel Tomančák. "Elastic volume reconstruction from series of ultra-thin microscopy sections". In: *Nature methods* 9.7 (2012), p. 717.

[105] Dominik Scherer, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: *International conference on artificial neural networks*. Springer. 2010, pp. 92–101.

[106] Martin Schiegg, Philipp Hanslovsky, Carsten Haubold, Ullrich Koethe, Lars Hufnagel, and Fred A Hamprecht. "Graphical model for joint segmentation and tracking of multiple dividing cells". In: *Bioinformatics* 31.6 (2014), pp. 948–956.

[107] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. "Fiji: an open-source platform for biological-image analysis". In: *Nature methods* 9.7 (2012), p. 676.

[108] Johannes Schindelin, Curtis T Rueden, Mark C Hiner, and Kevin W Eliceiri. "The ImageJ ecosystem: An open platform for biomedical image analysis". In: *Molecular reproduction and development* 82.7-8 (2015), pp. 518–529.

[109] Uwe Schmidt, Martin Weigert, Coleman Broaddus, and Gene Myers. "Cell Detection with Star-convex Polygons". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2018, pp. 265–273.

[110] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. "NIH Image to ImageJ: 25 years of image analysis". In: *Nature methods* 9.7 (2012), p. 671.

[111] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. "Robust object recognition with cortex-like mechanisms". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 3 (2007), pp. 411–426.

[112] H Sebastian Seung. "Reading the book of memory: sparse sampling versus dense mapping of connectomes". In: *Neuron* 62.1 (2009), pp. 17–29.

[113] Mojtaba Seyedhosseini, Mehdi Sajjadi, and Tolga Tasdizen. "Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 2168–2175.

[114] Arlo Sheridan et al. "Local Shape Descriptors for Neuron Segmentation". in preparation.

[115] Wenzhe Shi, Jose Caballero, Lucas Theis, Ferenc Huszar, Andrew Aitken, Christian Ledig, and Zehan Wang. "Is the deconvolution layer the same as a convolutional layer?" In: *arXiv preprint arXiv:1609.07009* (2016).

[116] Christoph Sommer, Christoph Straehle, Ullrich Koethe, and Fred A Hamprecht. "Ilastik: Interactive learning and segmentation toolkit". In: *2011 IEEE international symposium on biomedical imaging: From nano to macro*. IEEE. 2011, pp. 230–233.

[117] Olaf Sporns, Giulio Tononi, and Rolf Kötter. "The human connectome: a structural description of the human brain". In: *PLoS computational biology* 1.4 (2005), e42.

[118] Jon Sporring, Mahdieh Khanmohammadi, Sune Darkner, Nicoletta Nava, et al. "Estimating the thickness of ultra thin sections for electron microscopy by image statistics". In: *ISBI'14*. Apr. 2014, pp. 157–160. DOI: 10.1109/ISBI.2014.6867833.

[119] Benjamin Titze and Christel Genoud. "Volume scanning electron microscopy for imaging biological ultrastructure". In: *Biology of the Cell* 108.11 (2016), pp. 307–323.

[120] Srinivas C Turaga, Kevin L Briggman, Moritz Helmstaedter, Winfried Denk, and H Sebastian Seung. "Maximin affinity learning of image segmentation". In: *arXiv preprint arXiv:0911.5372* (2009).

[121] Srinivas C Turaga, Joseph F Murray, Viren Jain, Fabian Roth, Moritz Helmstaedter, Kevin Briggman, Winfried Denk, and H Sebastian Seung. "Convolutional networks can learn to generate affinity graphs for image segmentation". In: *Neural computation* 22.2 (2010), pp. 511–538.

[122] Mustafa Gokhan Uzunbas, Chao Chen, and Dimitris Metaxas. "An efficient conditional random field approach for automatic and interactive neuron segmentation". In: *Medical image analysis* 27 (2016), pp. 31–44.

[123] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation". In: *Computing in Science & Engineering* 13.2 (2011), p. 22.

[124] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. "scikit-image: image processing in Python". In: *PeerJ* 2 (2014), e453.

[125] Amelio Vazquez-Reina, Michael Gelbart, Daniel Huang, Jeff Lichtman, Eric Miller, and Hanspeter Pfister. "Segmentation fusion for connectomics". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 177–184.

[126] B. F. Voigt. "Der Handlungsreisende, wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein". In: *Commis-Voageur, Ilmenau* (1831).

[127] Ching-Wei Wang, Eric Budiman Gosno, and Yen-Sheng Li. "Fully automatic and robust 3D registration of serial-section microscopic images". In: *Scientific reports* 5 (2015), p. 15051.

[128] John G White, Eileen Southgate, J Nichol Thomson, and Sydney Brenner. "The structure of the nervous system of the nematode Caenorhabditis elegans". In: *Philos Trans R Soc Lond B Biol Sci* 314.1165 (1986), pp. 1–340.

[129] Bernard Widrow and Michael A Lehr. "30 years of adaptive neural networks: perceptron, madaline, and backpropagation". In: *Proceedings of the IEEE* 78.9 (1990), pp. 1415–1442.

[130] C Xu and H Hess. "A Closer Look at the Brain in 3D Using FIB-SEM". In: *Microscopy and Microanalysis* 17.Supplement S2 (July 2011), pp. 664–665. ISSN: 1435-8115. DOI: 10.1017/S1431927611004193.

[131] Julian Yarkony, Alexander Ihler, and Charless C Fowlkes. "Fast planar correlation clustering for image segmentation". In: *European Conference on Computer Vision*. Springer. 2012, pp. 568–581.

[132] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *CoRR* abs/1511.07122 (2016).

[133] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster Computing with Working Sets". In: *Proceedings of the 2nd USENIX Conference on Hot topics in cloud computing*. HotCloud'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 10–10.

[134] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[135] Jingyi Zhang, Beverly AS Reyes, Jennifer A Ross, Victoria Trovillion, and Elisabeth J Van Bockstaele. "Advances in Neuroscience Using Transmission Electron Microscopy: A Historical Perspective". In: *Transmission Electron Microscopy Methods for Understanding the Brain*. Springer, 2016, pp. 1–20.

[136] Ting Zhao, Donald J Olbris, Yang Yu, and Stephen M Plaza. "NeuTu: Software for Collaborative, Large-Scale, Segmentation-Based Connectome Reconstruction". In: *Frontiers in Neural Circuits* 12 (2018).

[137] Zhihao Zheng, J Scott Lauritzen, Eric Perlman, Camenzind G Robinson, Matthew Nichols, Daniel Milkie, Omar Torrens, John Price, Corey B Fisher, Nadiya Sharifi, et al. "A complete electron microscopy volume of the brain of adult Drosophila melanogaster". In: *Cell* 174.3 (2018), pp. 730–743.

Appendices

A

ARTIFACT CORRECTION

A.1 PARAMETERS

We make use of an alternating least squares scheme for solving equation (16) More explicitly, we repeatedly solve

$$\bar{s}^* = \arg\min_{\bar{s}} \mathrm{SSE}_{\mathrm{fit}} \tag{98}$$

$$\mathbf{m}^* = \arg\min_{\mathbf{m}} \mathrm{SSE}_{\mathrm{assess}} \tag{99}$$

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \mathrm{SSE}_{\mathrm{shift}} \tag{100}$$

equations (98) to (100) in sequence with two of three variables ($\bar{s}$, $\mathbf{m}$, $\mathbf{c}$) fixed to the current value until convergence or a user-specified number of iterations is reached. This alternating least squares approach requires a set of user defined parameters that are defined in table 14, along with default values. Only the most important parameters are exposed in the Fiji "Z-Spacing Correction" plugin, as indicated in table 14.

For non-planar distortion estimation, locality and resolution, as well as inference parameters for each stage need to be specified. A config file in the *JSON* format holds these parameters along with meta information, such as input and output paths. The list defined at the key "options" holds a *JSON* object of the parameters for each stage. The "steps" parameter defines the spacing in x and y between local estimates. The value of "steps" should be decreasing with each stage for an increasingly fine grid of estimates. The "radii" define the size of the field of view for similarity measure calculation of the local estimates and should vary approximately proportional to "steps", as an increasing resolution of the grid of estimates requires an increased locality of each estimate. Lastly, the "inference" parameters define the inference options (table 14) at each stage. If an option is not specified, it defaults to the value of the previous iteration, or–for the first stage–to the respective default value listed in table 14.

The parameters used for the non-planar distortion experiments in section 2.4.2.3 are listed in listings A.1 and A.2, omitting any meta information to avoid unnecessary clutter.

Table 14: Parameters for planar section spacing estimation.

Parameter		Exposed in Fiji as	Default
comparisonRange	Consider only as many neighbors in similarity curve.	test maximally	10
nIterations	Number of iterations for alternating least squares optimization (eq. 16 in the main text).	outer iterations	100
shiftProportion	Introduce damping by applying only proportional shift at each iteration of optimization.	outer regularization $1 - \text{shiftProportion}$	0.6
scalingFactorEstimationIterations	Number of iterations for optimization of scaling factors $\mathbf{m}$ (equation (99)).	inner iterations	10
scalingFactorRegularizerWeight	Element-wise regularization of $\mathbf{m}$ towards 1.	inner regularization	0.1
withReorder	Allow re-ordering of sections if set to *true*.	allow reordering	*true*
minimumSectionThickness	Minimum distance between sections if *withReorder* is set to *false* and sections are re-ordered in the estimate. The initial order is kept and the spacing is set to *minimumSectionThickness*.	—	0.01
coordinateUpdateRegularizerWeight	Element-wise regularization towards initial coordinates	—	0
regularizationType	Confine estimate to initial interval (*BORDER*), map linearly onto starting coordinates (*IDENTITY*), or do not regularize at all (*NONE*).	—	*BORDER*
forceMonotonicity	Ignore similarity measurements that break assumption of monotonicity. In practice this happens only for distant section pairs.	—	*false*
estimateWindowRadius	Spacing and field of view for estimates of $\bar{s}_i$. Use complete field of view if negative.	—	-1

Listing A.1: Parameters used in Experiment 2.4.2.3 (a).

```
{
  "options": [
    {
      "steps": [4000, 2500],
      "radii": [2001, 1251],
    },
    {
      "steps": [2000, 1250],
      "radii": [1001, 626],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.1,
        "comparisonRange": 15,
        "nIterations": 1000
      }
    },
    {
      "steps": [1000,625],
      "radii": [666, 416],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.1,
        "comparisonRange": 15,
        "nIterations": 1000
      }
    },
    {
      "steps": [500, 312],
      "radii": [500, 300],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.2,
        "comparisonRange": 15,
        "nIterations": 200
      }
    },
    {
      "steps": [250, 156],
      "radii": [300, 300],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.2,
        "comparisonRange": 15,
        "nIterations": 100
      }
    },
    {
      "steps": [125, 78],
      "radii": [200, 200],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.3,
        "comparisonRange": 10
      }
    },
    {
      "steps": [62, 39],
      "radii": [100, 100],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.4,
        "comparisonRange": 10,
        "nIterations": 50
      }
```

```
    },
    {
      "steps": [31, 19],
      "radii": [50, 50],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.5,
        "comparisonRange": 5,
        "nIterations": 10
      }
    }
  ]
}
```

Listing A.2: Parameters used in Experiment 2.4.2.3 (b).

```
{
  "options": [
    {
      "steps": [3000, 475],
      "radii": [1500, 474],
    },
    {
      "steps": [1500, 475],
      "radii": [1000, 474],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.1,
        "comparisonRange": 15,
        "nIterations": 100
      }
    },
    {
      "steps": [750, 474],
      "radii": [666, 416],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.1,
        "comparisonRange": 15,
        "nIterations": 100
      }
    },
    {
      "steps": [500, 312],
      "radii": [500, 300],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.2,
        "comparisonRange": 15,
        "nIterations": 20
      }
    },
    {
      "steps": [250, 156],
      "radii": [300,300],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.2,
        "comparisonRange": 15,
        "nIterations": 10
      }
    },
    {
      "steps": [125,78],
      "radii": [200, 200],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.3,
        "comparisonRange": 10
      }
    },
    {
      "steps": [62, 39],
      "radii": [100, 100],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.4,
        "comparisonRange": 10,
        "nIterations": 10
      }
```

```
    },
    {
      "steps": [31, 19],
      "radii": [50, 50],
      "inference": {
        "regularizationType": "NONE",
        "coordinateUpdateRegularizerWeight": 0.5,
        "comparisonRange": 5,
        "nIterations": 10
      }
    }
  ]
}
```

B

RECONSTRUCTION

B.1 PREDICTION AND RECONSTRUCITON EXAMPLES

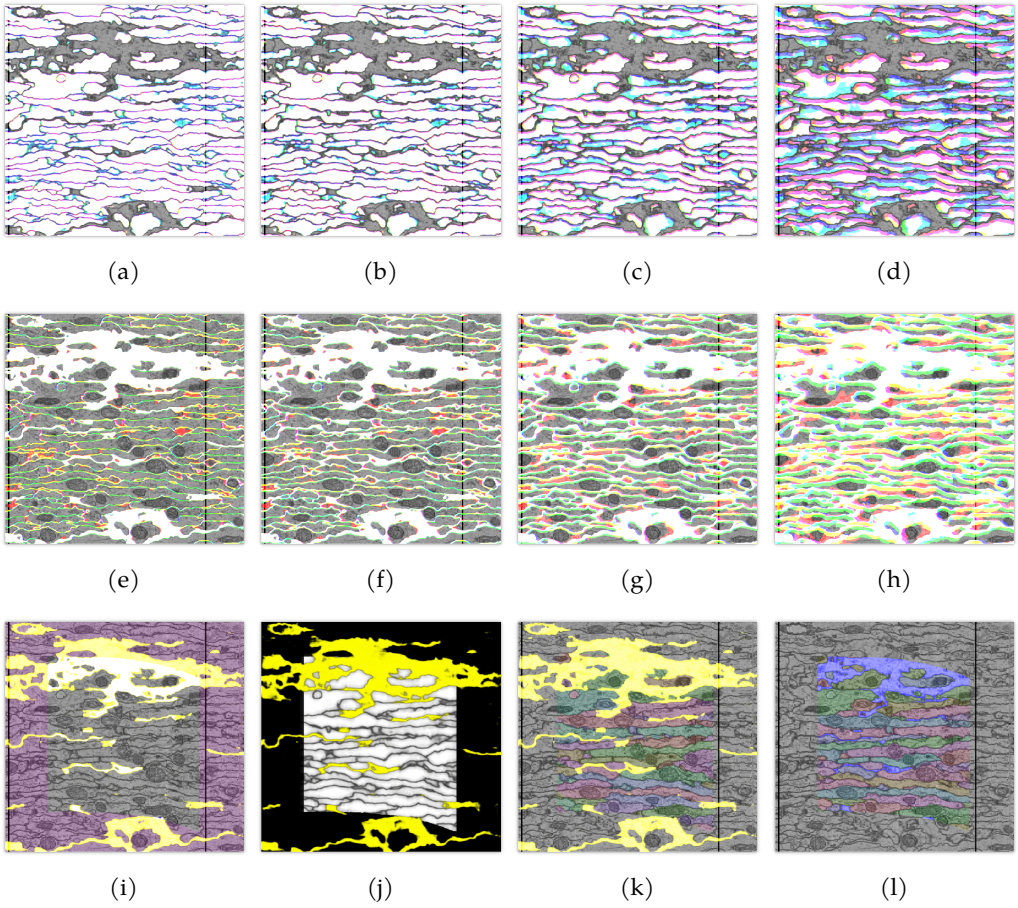


Figure 59: Cross-section of prediction and reconstruction for sample A with best performing network qi-mse-ng (table 9): The horizontal (x)/vertical (y) image axes are aligned with the third/second axis of the data. The corresponding resolution ($x \times y$)is 40nm × 4nm for raw data, and $^{40}/_{3}$nm × 12nm for interpolated ground truth, predictions and reconstruction. The annotated area spans 5×10^3 nm $= 5\,\mu$m from left to right (enclosed by magenta shading (i)). Each set of affinity channels, a_1, a_5, a_9(a), a_2, a_6, a_{10}(b), a_3, a_7, a_{11}(c), and a_4, a_8, a_{12}(d), is projected onto RGB channels with contrast range $[0, 1]$. The inverted affinities (e)–(h) are displayed with contrast range $[0.5, 0.7]$. Yellow glia predictions g with contrast range $[0, 1]$ are overlaid with blue ground truth in (i) to highlight false negative (blue), false positive (yellow), and correct predictions (white) within the masked area. Glia annotations are not available outside the annotated and yellow predictions cannot be considered false there. The averaged and scaled affinities $\bar{a}\left(1 - g_0^1\right)$ are restricted to the annotated volume (j). Predicted glia voxels are not considered for neuron reconstruction (k). The ground truth (l) is shown for comparison and with glia highlighted in blue color.

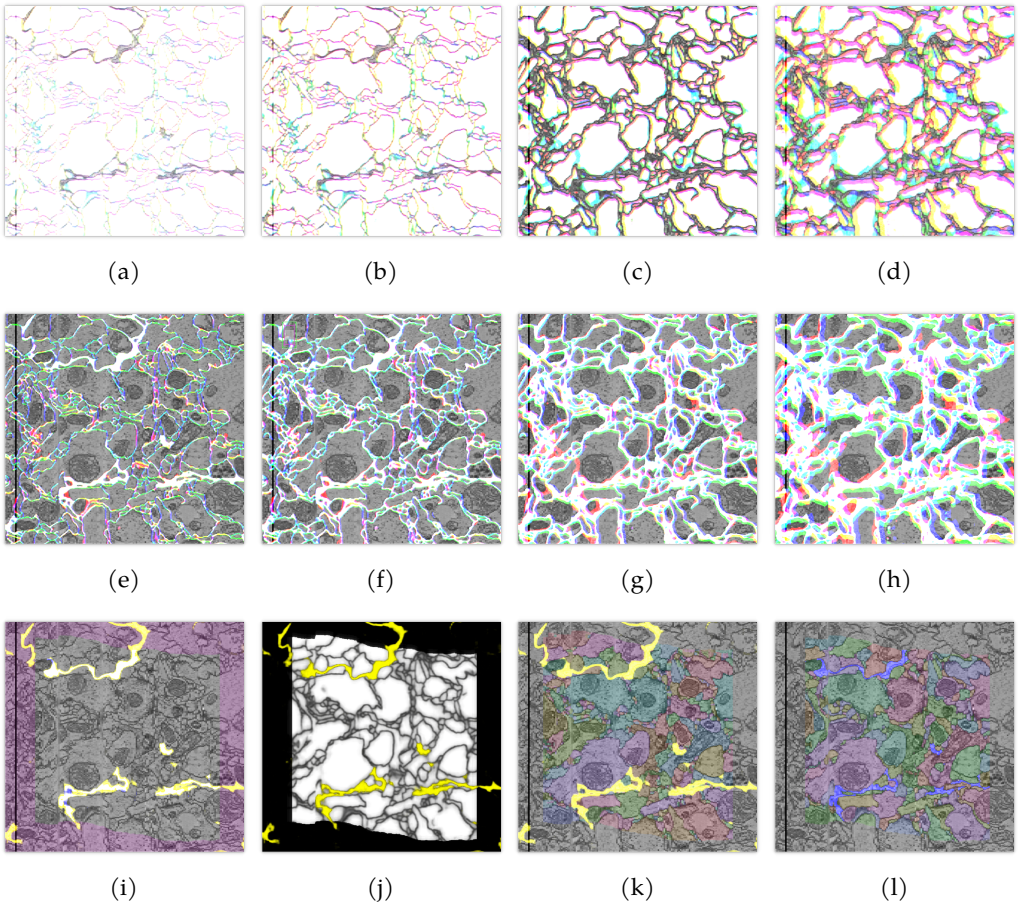


Figure 60: Cross-section of prediction and reconstruction for sample B with best performing network qi-mse-ng (table 9): The horizontal (x) and vertical (y) image axes are aligned with the third and second axis of the data, respectively. The corresponding resolution ($x \times y$)is 40nm × 4nm for raw data, and $^{40}/_{3}$nm × 12nm for interpolated ground truth, predictions and reconstruction. The annotated area spans 5×10^3 nm $= 5\,\mu$m from left to right (enclosed by magenta shading (i)). Each set of affinity channels, a_1, a_5, a_9(a), a_2, a_6, a_{10}(b), a_3, a_7, a_{11}(c), and a_4, a_8, a_{12}(d), is projected onto RGB channels with contrast range $[0, 1]$. The inverted affinities (e)–(h) are displayed with contrast range $[0.5, 0.7]$. Yellow glia predictions g with contrast range $[0, 1]$ are overlaid with blue ground truth in (i) to highlight false negative (blue), false positive (yellow), and correct predictions (white) within the masked area. Glia annotations are not available outside the annotated and yellow predictions cannot be considered false there. The averaged and scaled affinities $\bar{a}\left(1 - g_0^1\right)$ are restricted to the annotated volume (j). Predicted glia voxels are not considered for neuron reconstruction (k). The ground truth (l) is shown for comparison and with glia highlighted in blue color.

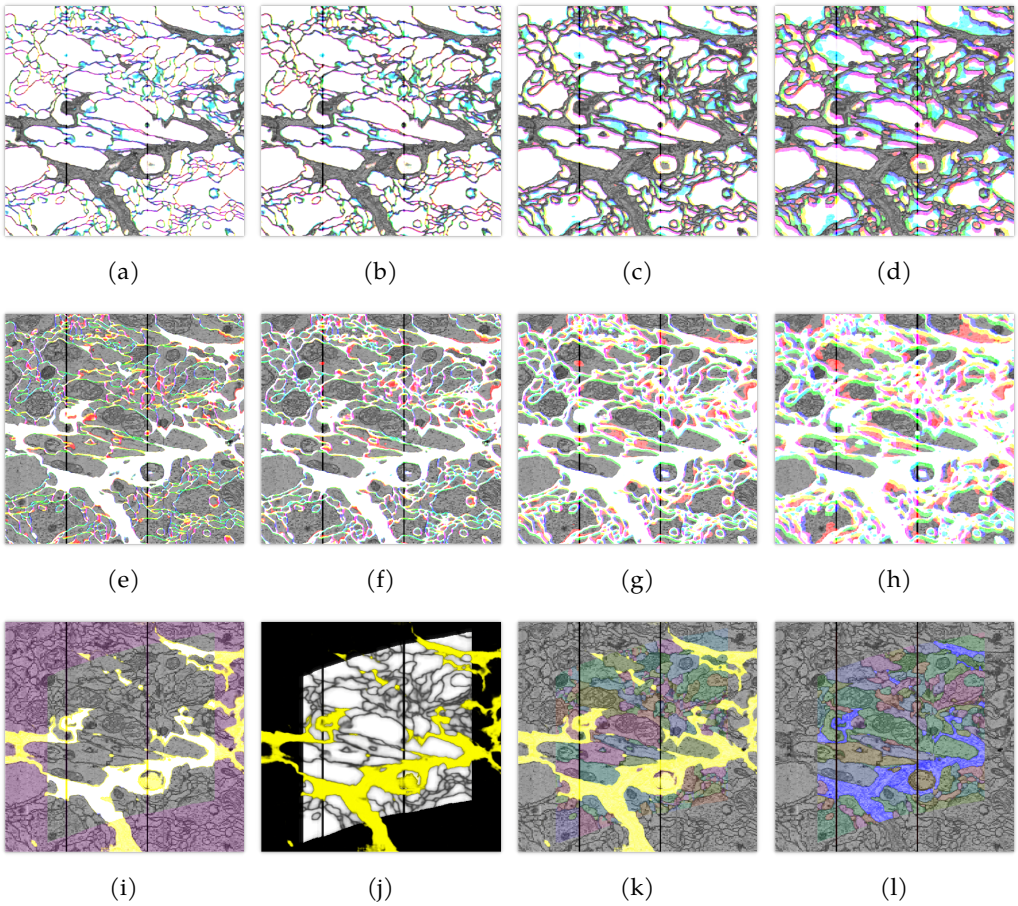


Figure 61: Cross-section of prediction and reconstruction for sample C with best performing network qi-mse-ng (table 9): The horizontal (x)/vertical (y) image axes are aligned with the third/second axis of the data. The corresponding resolution ($x \times y$)is 40nm × 4nm for raw data, and $^{40}/_{3}$nm × 12nm for interpolated ground truth, predictions and reconstruction. The annotated area spans 5×10^3 nm $= 5\,\mu$m from left to right (enclosed by magenta shading (i)). Each set of affinity channels, a_1, a_5, a_9(a), a_2, a_6, a_{10}(b), a_3, a_7, a_{11}(c), and a_4, a_8, a_{12}(d), is projected onto RGB channels with contrast range $[0, 1]$. The inverted affinities (e)–(h) are displayed with contrast range $[0.5, 0.7]$. Yellow glia predictions g with contrast range $[0, 1]$ are overlaid with blue ground truth in (i) to highlight false negative (blue), false positive (yellow), and correct predictions (white) within the masked area. Glia annotations are not available outside the annotated and yellow predictions cannot be considered false there. The averaged and scaled affinities $\bar{a}\,(1 - g_0^1)$ are restricted to the annotated volume (j). Predicted glia voxels are not considered for neuron reconstruction (k). The ground truth (l) is shown for comparison and with glia highlighted in blue color.

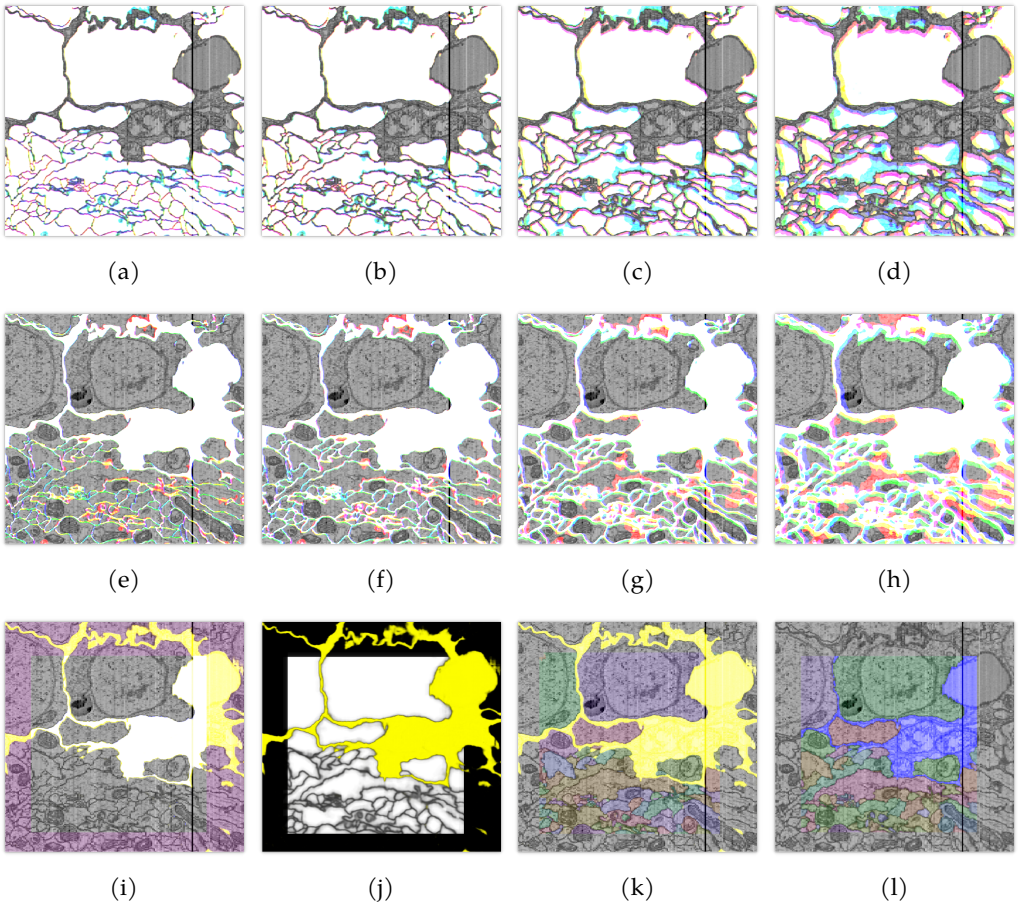


Figure 62: Cross-section of prediction and reconstruction for sample 0 with best performing network qi-mse-ng (table 9): The horizontal (x)/vertical (y) image axes are aligned with the third/second axis of the data. The corresponding resolution ($x \times y$)is 40nm × 4nm for raw data, and $^{40}/_{3}$nm × 12nm for interpolated ground truth, predictions and reconstruction. The annotated area spans 5×10^3 nm $= 5\,\mu$m from left to right (enclosed by magenta shading (i)). Each set of affinity channels, a_1, a_5, a_9(a), a_2, a_6, a_{10}(b), a_3, a_7, a_{11}(c), and a_4, a_8, a_{12}(d), is projected onto RGB channels with contrast range $[0, 1]$. The inverted affinities (e)–(h) are displayed with contrast range $[0.5, 0.7]$. Yellow glia predictions g with contrast range $[0, 1]$ are overlaid with blue ground truth in (i) to highlight false negative (blue), false positive (yellow), and correct predictions (white) within the masked area. Glia annotations are not available outside the annotated and yellow predictions cannot be considered false there. The averaged and scaled affinities $\bar{a}\left(1 - g_0^1\right)$ are restricted to the annotated volume (j). Predicted glia voxels are not considered for neuron reconstruction (k). The ground truth (l) is shown for comparison and with glia highlighted in blue color.

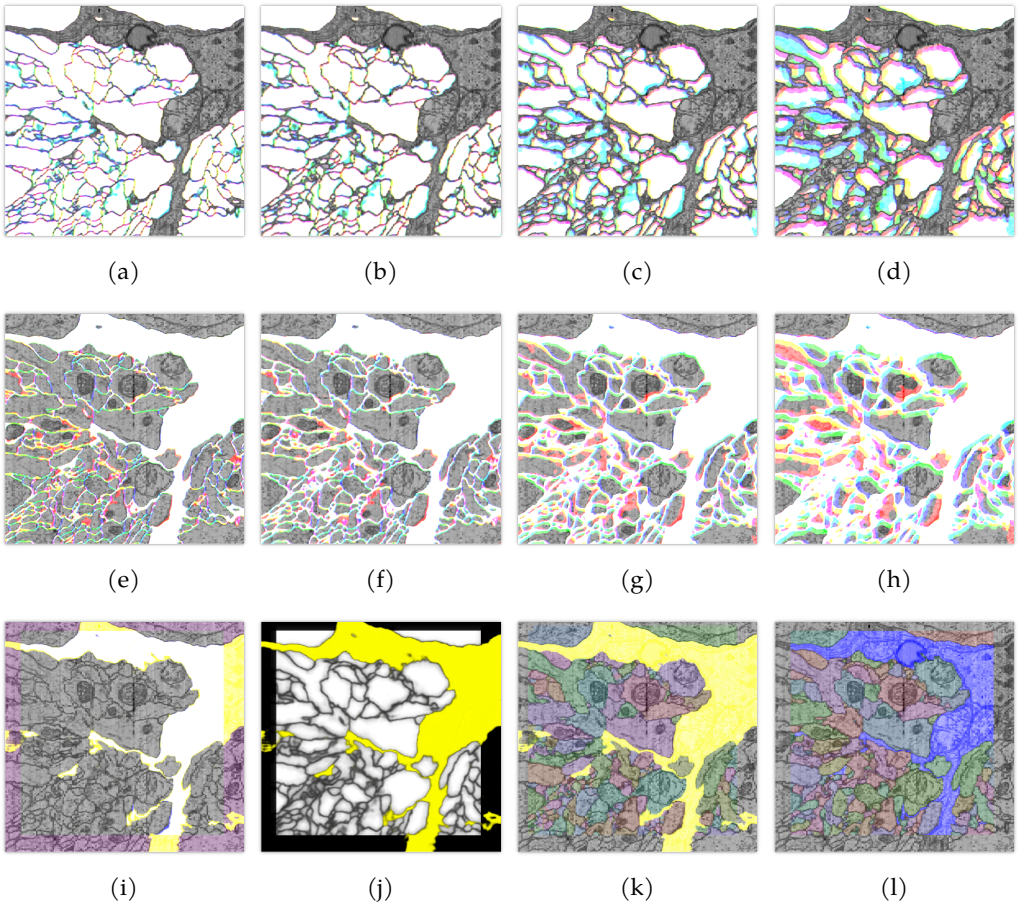


Figure 63: Cross-section of prediction and reconstruction for sample 2 with best performing network qi-mse-ng (table 9): The horizontal (x)/vertical (y) image axes are aligned with the third/second axis of the data. The corresponding resolution ($x \times y$)is 40nm × 4nm for raw data, and $^{40}/_{3}$nm × 12nm for interpolated ground truth, predictions and reconstruction. The annotated area spans 5×10^3 nm $= 5\,\mu$m from left to right (enclosed by magenta shading (i)). Each set of affinity channels, a_1, a_5, a_9(a), a_2, a_6, a_{10}(b), a_3, a_7, a_{11}(c), and a_4, a_8, a_{12}(d), is projected onto RGB channels with contrast range $[0, 1]$. The inverted affinities (e)–(h) are displayed with contrast range $[0.5, 0.7]$. Yellow glia predictions g with contrast range $[0, 1]$ are overlaid with blue ground truth in (i) to highlight false negative (blue), false positive (yellow), and correct predictions (white) within the masked area. Glia annotations are not available outside the annotated and yellow predictions cannot be considered false there. The averaged and scaled affinities $\bar{a}\,(1 - g_0^1)$ are restricted to the annotated volume (j). Predicted glia voxels are not considered for neuron reconstruction (k). The ground truth (l) is shown for comparison and with glia highlighted in blue color.

B.2 PERFORMANCE EVALUATION OF ALL PARAMETER SETS

Table 15: VOI_s performance on 25% of the data that were not used for training for all architectures and parameter sets. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging.

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	—	0.189	0.004	0.001	0.777	0.032	0.301	0.018
qi-mse	0.1	0.1	0.148	0.014	0.054	0.593	0.122	0.072	0.034
qi-mse	0.1	0.2	0.136	0.009	0.037	0.585	0.113	0.054	0.021
qi-mse	0.1	0.3	0.132	0.008	0.029	0.582	0.109	0.047	0.015
qi-mse	0.1	0.4	0.129	0.008	0.024	0.583	0.106	0.044	0.012
qi-mse	0.1	0.5	0.128	0.008	0.02	0.585	0.104	0.043	0.01
qi-mse	0.1	0.6	0.128	0.008	0.018	0.588	0.102	0.044	0.01
qi-mse	0.1	0.7	0.128	0.008	0.016	0.592	0.101	0.044	0.009
qi-mse	0.1	0.8	0.129	0.009	0.014	0.598	0.099	0.046	0.009
qi-mse	0.1	0.9	0.133	0.01	0.014	0.613	0.097	0.052	0.009
qi-mse	0.3	—	0.385	0.059	0.068	1.264	0.116	0.739	0.066
qi-mse	0.3	0.1	0.165	0.017	0.076	0.604	0.176	0.075	0.042
qi-mse	0.3	0.2	0.154	0.012	0.061	0.595	0.168	0.056	0.029
qi-mse	0.3	0.3	0.149	0.011	0.053	0.594	0.164	0.051	0.023
qi-mse	0.3	0.4	0.148	0.011	0.049	0.596	0.161	0.049	0.021
qi-mse	0.3	0.5	0.148	0.011	0.046	0.598	0.16	0.049	0.021
qi-mse	0.3	0.6	0.148	0.012	0.044	0.601	0.159	0.052	0.02
qi-mse	0.3	0.7	0.151	0.013	0.044	0.608	0.159	0.061	0.02
qi-mse	0.3	0.8	0.154	0.014	0.043	0.62	0.158	0.071	0.02
qi-mse	0.3	0.9	0.165	0.017	0.045	0.643	0.164	0.098	0.021
qi-mse	0.5	—	0.88	0.252	0.61	2.183	0.43	1.186	0.617
qi-mse	0.5	0.1	0.435	0.128	0.457	0.985	0.387	0.198	0.456
qi-mse	0.5	0.2	0.427	0.124	0.437	0.981	0.379	0.191	0.45
qi-mse	0.5	0.3	0.43	0.123	0.461	0.983	0.376	0.189	0.449
qi-mse	0.5	0.4	0.433	0.124	0.483	0.979	0.374	0.193	0.447
qi-mse	0.5	0.5	0.437	0.125	0.487	0.984	0.375	0.201	0.454
qi-mse	0.5	0.6	0.447	0.125	0.491	1.008	0.386	0.219	0.454
qi-mse	0.5	0.7	0.452	0.127	0.494	1.021	0.385	0.231	0.454
qi-mse	0.5	0.8	0.458	0.129	0.496	1.035	0.387	0.245	0.457
qi-mse	0.5	0.9	0.472	0.132	0.505	1.066	0.391	0.271	0.464
qi-mse	0.7	—	1.257	0.509	0.959	2.62	0.828	1.527	1.097
qi-mse	0.7	0.1	0.74	0.338	0.853	1.267	0.758	0.326	0.898
qi-mse	0.7	0.2	0.738	0.335	0.847	1.27	0.756	0.327	0.894
qi-mse	0.7	0.3	0.742	0.341	0.848	1.273	0.758	0.336	0.895
qi-mse	0.7	0.4	0.748	0.342	0.848	1.296	0.759	0.347	0.896
qi-mse	0.7	0.5	0.751	0.343	0.847	1.302	0.761	0.357	0.897
qi-mse	0.7	0.6	0.756	0.344	0.852	1.311	0.762	0.369	0.898
qi-mse	0.7	0.7	0.764	0.345	0.86	1.322	0.771	0.382	0.902
qi-mse	0.7	0.8	0.773	0.347	0.864	1.341	0.785	0.398	0.905
qi-mse	0.7	0.9	0.789	0.351	0.872	1.375	0.793	0.432	0.909
qi-mse	0.9	—	3.107	4.089	2.255	3.773	2.94	2.675	2.912
qi-mse	0.9	0.1	2.539	3.911	2.165	2.298	2.715	1.375	2.769
qi-mse	0.9	0.2	2.542	3.912	2.164	2.302	2.715	1.389	2.772
qi-mse	0.9	0.3	2.551	3.917	2.171	2.307	2.729	1.4	2.778
qi-mse	0.9	0.4	2.555	3.921	2.174	2.313	2.733	1.405	2.782
qi-mse	0.9	0.5	2.559	3.922	2.175	2.323	2.735	1.416	2.784
qi-mse	0.9	0.6	2.571	3.927	2.178	2.332	2.739	1.461	2.789
qi-mse	0.9	0.7	2.579	3.929	2.184	2.343	2.742	1.48	2.796
qi-mse	0.9	0.8	2.589	3.935	2.189	2.359	2.749	1.503	2.8
qi-mse	0.9	0.9	2.608	3.943	2.196	2.398	2.768	1.537	2.808
qi-mse-ng	0.1	—	0.16	0.005	0.0	0.763	0.019	0.166	0.006
qi-mse-ng	0.1	0.1	0.132	0.009	0.037	0.624	0.017	0.072	0.03
qi-mse-ng	0.1	0.2	0.124	0.007	0.026	0.618	0.009	0.061	0.02
qi-mse-ng	0.1	0.3	0.12	0.006	0.021	0.616	0.006	0.057	0.015
qi-mse-ng	0.1	0.4	0.118	0.006	0.017	0.616	0.004	0.055	0.011
qi-mse-ng	0.1	0.5	0.117	0.006	0.014	0.618	0.003	0.055	0.008
qi-mse-ng	0.1	0.6	0.117	0.006	0.011	0.62	0.003	0.056	0.007
qi-mse-ng	0.1	0.7	0.117	0.006	0.01	0.624	0.003	0.056	0.006
qi-mse-ng	0.1	0.8	0.119	0.006	0.009	0.632	0.004	0.06	0.005
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 15 (VOI_s on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse-ng	0.1	0.9	0.126	0.006	0.01	0.651	0.007	0.074	0.006
qi-mse-ng	0.3	—	0.332	0.036	0.039	1.261	0.105	0.527	0.025
qi-mse-ng	0.3	0.1	0.134	0.01	0.042	0.626	0.019	0.075	0.032
qi-mse-ng	0.3	0.2	0.126	0.007	0.031	0.62	0.012	0.066	0.022
qi-mse-ng	0.3	0.3	0.123	0.007	0.025	0.619	0.009	0.062	0.017
qi-mse-ng	0.3	0.4	0.121	0.007	0.022	0.619	0.007	0.06	0.014
qi-mse-ng	0.3	0.5	0.121	0.007	0.018	0.622	0.006	0.06	0.012
qi-mse-ng	0.3	0.6	0.121	0.007	0.016	0.625	0.006	0.062	0.011
qi-mse-ng	0.3	0.7	0.124	0.007	0.016	0.635	0.007	0.067	0.01
qi-mse-ng	0.3	0.8	0.129	0.007	0.016	0.654	0.008	0.079	0.01
qi-mse-ng	0.3	0.9	0.139	0.008	0.017	0.687	0.011	0.101	0.011
qi-mse-ng	0.5	—	0.693	0.186	0.358	1.997	0.269	0.892	0.456
qi-mse-ng	0.5	0.1	0.324	0.09	0.27	0.954	0.134	0.152	0.343
qi-mse-ng	0.5	0.2	0.321	0.09	0.263	0.952	0.128	0.153	0.337
qi-mse-ng	0.5	0.3	0.309	0.087	0.26	0.891	0.125	0.156	0.333
qi-mse-ng	0.5	0.4	0.321	0.087	0.269	0.958	0.124	0.156	0.331
qi-mse-ng	0.5	0.5	0.323	0.087	0.269	0.963	0.123	0.166	0.33
qi-mse-ng	0.5	0.6	0.326	0.087	0.272	0.97	0.123	0.174	0.331
qi-mse-ng	0.5	0.7	0.334	0.09	0.279	0.989	0.124	0.19	0.333
qi-mse-ng	0.5	0.8	0.334	0.091	0.281	0.95	0.125	0.211	0.344
qi-mse-ng	0.5	0.9	0.349	0.092	0.291	0.992	0.13	0.245	0.347
qi-mse-ng	0.7	—	1.143	0.377	0.868	2.48	0.749	1.356	1.028
qi-mse-ng	0.7	0.1	0.652	0.252	0.748	1.256	0.496	0.313	0.845
qi-mse-ng	0.7	0.2	0.654	0.25	0.751	1.249	0.52	0.314	0.837
qi-mse-ng	0.7	0.3	0.651	0.249	0.752	1.26	0.488	0.319	0.838
qi-mse-ng	0.7	0.4	0.659	0.249	0.752	1.267	0.528	0.326	0.834
qi-mse-ng	0.7	0.5	0.664	0.25	0.754	1.283	0.529	0.336	0.835
qi-mse-ng	0.7	0.6	0.669	0.25	0.754	1.292	0.529	0.348	0.838
qi-mse-ng	0.7	0.7	0.677	0.25	0.757	1.308	0.531	0.373	0.841
qi-mse-ng	0.7	0.8	0.689	0.251	0.763	1.329	0.538	0.405	0.846
qi-mse-ng	0.7	0.9	0.71	0.26	0.773	1.378	0.546	0.449	0.852
qi-mse-ng	0.9	—	2.658	3.306	1.905	3.486	2.415	2.354	2.483
qi-mse-ng	0.9	0.1	2.106	3.125	1.813	2.133	2.138	1.143	2.284
qi-mse-ng	0.9	0.2	2.111	3.13	1.815	2.139	2.142	1.157	2.284
qi-mse-ng	0.9	0.3	2.115	3.133	1.821	2.147	2.141	1.168	2.283
qi-mse-ng	0.9	0.4	2.123	3.139	1.821	2.165	2.142	1.186	2.286
qi-mse-ng	0.9	0.5	2.128	3.139	1.822	2.172	2.144	1.2	2.29
qi-mse-ng	0.9	0.6	2.137	3.147	1.827	2.187	2.147	1.218	2.296
qi-mse-ng	0.9	0.7	2.146	3.157	1.832	2.2	2.15	1.236	2.299
qi-mse-ng	0.9	0.8	2.156	3.162	1.837	2.223	2.153	1.262	2.301
qi-mse-ng	0.9	0.9	2.181	3.169	1.843	2.271	2.178	1.31	2.314
qi-mls-pre	0.1	—	1.018	1.015	0.756	1.853	0.649	0.855	0.983
qi-mls-pre	0.1	0.1	0.872	0.941	0.758	1.403	0.69	0.592	0.846
qi-mls-pre	0.1	0.2	0.871	0.932	0.742	1.401	0.725	0.597	0.829
qi-mls-pre	0.1	0.3	0.872	0.931	0.728	1.405	0.719	0.625	0.824
qi-mls-pre	0.1	0.4	0.869	0.931	0.719	1.412	0.717	0.614	0.819
qi-mls-pre	0.1	0.5	0.885	0.931	0.712	1.421	0.724	0.703	0.819
qi-mls-pre	0.1	0.6	0.889	0.928	0.707	1.432	0.724	0.725	0.817
qi-mls-pre	0.1	0.7	0.893	0.928	0.703	1.445	0.725	0.741	0.817
qi-mls-pre	0.1	0.8	0.908	0.929	0.716	1.477	0.73	0.777	0.816
qi-mls-pre	0.1	0.9	0.935	0.93	0.721	1.527	0.749	0.867	0.816
qi-mls-pre	0.3	—	5.798	6.008	5.403	6.094	5.575	5.851	5.855
qi-mls-pre	0.3	0.1	5.349	5.848	5.265	5.067	5.292	4.956	5.667
qi-mls-pre	0.3	0.2	5.354	5.846	5.262	5.085	5.295	4.975	5.663
qi-mls-pre	0.3	0.3	5.36	5.846	5.26	5.098	5.296	4.996	5.663
qi-mls-pre	0.3	0.4	5.365	5.845	5.257	5.112	5.297	5.017	5.661
qi-mls-pre	0.3	0.5	5.371	5.846	5.25	5.128	5.3	5.04	5.66
qi-mls-pre	0.3	0.6	5.379	5.846	5.248	5.146	5.303	5.07	5.66
qi-mls-pre	0.3	0.7	5.389	5.846	5.247	5.171	5.307	5.103	5.66
qi-mls-pre	0.3	0.8	5.404	5.847	5.246	5.204	5.315	5.155	5.66
qi-mls-pre	0.3	0.9	5.433	5.848	5.246	5.268	5.332	5.247	5.661
qi-mls-pre	0.5	—	6.272	6.224	6.013	6.536	5.826	6.847	6.186
qi-mls-pre	0.5	0.1	5.723	6.022	5.8	5.436	5.457	5.661	5.965
qi-mls-pre	0.5	0.2	5.728	6.02	5.796	5.452	5.459	5.681	5.96
qi-mls-pre	0.5	0.3	5.732	6.019	5.792	5.465	5.459	5.7	5.959
qi-mls-pre	0.5	0.4	5.737	6.019	5.789	5.478	5.46	5.721	5.957
qi-mls-pre	0.5	0.5	5.743	6.019	5.786	5.495	5.462	5.743	5.956
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 15 (VOI_s on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre	0.5	0.6	5.751	6.019	5.784	5.513	5.465	5.773	5.955
qi-mls-pre	0.5	0.7	5.761	6.019	5.782	5.537	5.468	5.806	5.955
qi-mls-pre	0.5	0.8	5.776	6.02	5.781	5.57	5.476	5.857	5.955
qi-mls-pre	0.5	0.9	5.805	6.021	5.78	5.634	5.493	5.948	5.955
qi-mls-pre	0.7	—	6.272	6.224	6.013	6.536	5.826	6.847	6.186
qi-mls-pre	0.7	0.1	5.723	6.022	5.8	5.436	5.457	5.661	5.965
qi-mls-pre	0.7	0.2	5.728	6.02	5.796	5.452	5.459	5.681	5.96
qi-mls-pre	0.7	0.3	5.732	6.019	5.792	5.465	5.459	5.7	5.959
qi-mls-pre	0.7	0.4	5.737	6.019	5.789	5.478	5.46	5.721	5.957
qi-mls-pre	0.7	0.5	5.743	6.019	5.786	5.495	5.462	5.743	5.956
qi-mls-pre	0.7	0.6	5.751	6.019	5.784	5.513	5.465	5.773	5.955
qi-mls-pre	0.7	0.7	5.761	6.019	5.782	5.537	5.468	5.806	5.955
qi-mls-pre	0.7	0.8	5.776	6.02	5.781	5.57	5.476	5.857	5.955
qi-mls-pre	0.7	0.9	5.805	6.021	5.78	5.634	5.493	5.948	5.955
qi-mls-pre	0.9	—	6.272	6.224	6.013	6.536	5.826	6.847	6.186
qi-mls-pre	0.9	0.1	5.723	6.022	5.8	5.436	5.457	5.661	5.965
qi-mls-pre	0.9	0.2	5.728	6.02	5.796	5.452	5.459	5.681	5.96
qi-mls-pre	0.9	0.3	5.732	6.019	5.792	5.465	5.459	5.7	5.959
qi-mls-pre	0.9	0.4	5.737	6.019	5.789	5.478	5.46	5.721	5.957
qi-mls-pre	0.9	0.5	5.743	6.019	5.786	5.495	5.462	5.743	5.956
qi-mls-pre	0.9	0.6	5.751	6.019	5.784	5.513	5.465	5.773	5.955
qi-mls-pre	0.9	0.7	5.761	6.019	5.782	5.537	5.468	5.806	5.955
qi-mls-pre	0.9	0.8	5.776	6.02	5.781	5.57	5.476	5.857	5.955
qi-mls-pre	0.9	0.9	5.805	6.021	5.78	5.634	5.493	5.948	5.955
qi-mls-pre-ng	0.1	—	0.663	0.66	0.5	1.213	0.476	0.328	0.801
qi-mls-pre-ng	0.1	0.1	0.581	0.562	0.586	0.748	0.6	0.249	0.743
qi-mls-pre-ng	0.1	0.2	0.576	0.514	0.554	0.804	0.597	0.245	0.741
qi-mls-pre-ng	0.1	0.3	0.579	0.527	0.56	0.803	0.592	0.243	0.748
qi-mls-pre-ng	0.1	0.4	0.576	0.564	0.553	0.804	0.59	0.198	0.747
qi-mls-pre-ng	0.1	0.5	0.574	0.561	0.544	0.805	0.588	0.198	0.747
qi-mls-pre-ng	0.1	0.6	0.575	0.564	0.539	0.814	0.588	0.2	0.747
qi-mls-pre-ng	0.1	0.7	0.577	0.566	0.551	0.819	0.587	0.196	0.746
qi-mls-pre-ng	0.1	0.8	0.604	0.571	0.568	0.901	0.587	0.249	0.746
qi-mls-pre-ng	0.1	0.9	0.631	0.579	0.569	0.93	0.621	0.268	0.818
qi-mls-pre-ng	0.3	—	2.307	1.993	2.366	2.585	2.237	2.433	2.229
qi-mls-pre-ng	0.3	0.1	2.068	1.927	2.303	2.029	2.137	1.843	2.169
qi-mls-pre-ng	0.3	0.2	2.063	1.923	2.297	2.024	2.129	1.848	2.16
qi-mls-pre-ng	0.3	0.3	2.065	1.921	2.29	2.035	2.128	1.853	2.162
qi-mls-pre-ng	0.3	0.4	2.068	1.923	2.287	2.04	2.127	1.866	2.162
qi-mls-pre-ng	0.3	0.5	2.07	1.924	2.288	2.042	2.126	1.876	2.162
qi-mls-pre-ng	0.3	0.6	2.072	1.924	2.287	2.051	2.126	1.884	2.162
qi-mls-pre-ng	0.3	0.7	2.076	1.925	2.287	2.057	2.126	1.899	2.164
qi-mls-pre-ng	0.3	0.8	2.085	1.926	2.287	2.093	2.126	1.912	2.164
qi-mls-pre-ng	0.3	0.9	2.099	1.932	2.292	2.127	2.127	1.951	2.167
qi-mls-pre-ng	0.5	—	2.308	1.994	2.366	2.585	2.237	2.433	2.235
qi-mls-pre-ng	0.5	0.1	2.069	1.927	2.303	2.029	2.137	1.843	2.173
qi-mls-pre-ng	0.5	0.2	2.064	1.923	2.297	2.024	2.129	1.848	2.164
qi-mls-pre-ng	0.5	0.3	2.065	1.921	2.29	2.035	2.128	1.853	2.166
qi-mls-pre-ng	0.5	0.4	2.068	1.923	2.287	2.04	2.127	1.866	2.166
qi-mls-pre-ng	0.5	0.5	2.07	1.924	2.288	2.042	2.126	1.876	2.166
qi-mls-pre-ng	0.5	0.6	2.073	1.924	2.287	2.051	2.126	1.884	2.166
qi-mls-pre-ng	0.5	0.7	2.077	1.925	2.287	2.057	2.126	1.899	2.168
qi-mls-pre-ng	0.5	0.8	2.085	1.926	2.287	2.093	2.126	1.912	2.168
qi-mls-pre-ng	0.5	0.9	2.1	1.932	2.292	2.127	2.127	1.951	2.171
qi-mls-pre-ng	0.7	—	2.308	1.994	2.366	2.585	2.237	2.433	2.235
qi-mls-pre-ng	0.7	0.1	2.069	1.927	2.303	2.029	2.137	1.843	2.173
qi-mls-pre-ng	0.7	0.2	2.064	1.923	2.297	2.024	2.129	1.848	2.164
qi-mls-pre-ng	0.7	0.3	2.065	1.921	2.29	2.035	2.128	1.853	2.166
qi-mls-pre-ng	0.7	0.4	2.068	1.923	2.287	2.04	2.127	1.866	2.166
qi-mls-pre-ng	0.7	0.5	2.07	1.924	2.288	2.042	2.126	1.876	2.166
qi-mls-pre-ng	0.7	0.6	2.073	1.924	2.287	2.051	2.126	1.884	2.166
qi-mls-pre-ng	0.7	0.7	2.077	1.925	2.287	2.057	2.126	1.899	2.168
qi-mls-pre-ng	0.7	0.8	2.085	1.926	2.287	2.093	2.126	1.912	2.168
qi-mls-pre-ng	0.7	0.9	2.1	1.932	2.292	2.127	2.127	1.951	2.171
qi-mls-pre-ng	0.9	—	2.308	1.994	2.366	2.585	2.237	2.433	2.235
qi-mls-pre-ng	0.9	0.1	2.069	1.927	2.303	2.029	2.137	1.843	2.173
qi-mls-pre-ng	0.9	0.2	2.064	1.923	2.297	2.024	2.129	1.848	2.164
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 15 (VOI_s on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre-ng	0.9	0.3	2.065	1.921	2.29	2.035	2.128	1.853	2.166
qi-mls-pre-ng	0.9	0.4	2.068	1.923	2.287	2.04	2.127	1.866	2.166
qi-mls-pre-ng	0.9	0.5	2.07	1.924	2.288	2.042	2.126	1.876	2.166
qi-mls-pre-ng	0.9	0.6	2.073	1.924	2.287	2.051	2.126	1.884	2.166
qi-mls-pre-ng	0.9	0.7	2.077	1.925	2.287	2.057	2.126	1.899	2.168
qi-mls-pre-ng	0.9	0.8	2.085	1.926	2.287	2.093	2.126	1.912	2.168
qi-mls-pre-ng	0.9	0.9	2.1	1.932	2.292	2.127	2.127	1.951	2.171
qi-mls	0.1	—	1.189	0.837	1.074	1.973	1.116	0.865	1.27
qi-mls	0.1	0.1	0.979	0.799	1.037	1.423	1.002	0.445	1.166
qi-mls	0.1	0.2	0.978	0.808	1.035	1.438	0.998	0.431	1.16
qi-mls	0.1	0.3	0.98	0.81	1.034	1.443	0.995	0.438	1.16
qi-mls	0.1	0.4	0.98	0.818	1.032	1.446	0.997	0.439	1.149
qi-mls	0.1	0.5	0.983	0.818	1.031	1.453	1.007	0.442	1.147
qi-mls	0.1	0.6	0.986	0.818	1.033	1.466	0.999	0.447	1.154
qi-mls	0.1	0.7	0.99	0.824	1.036	1.473	0.997	0.456	1.158
qi-mls	0.1	0.8	1.001	0.824	1.043	1.485	1.006	0.491	1.159
qi-mls	0.1	0.9	1.017	0.829	1.048	1.511	1.013	0.538	1.164
qi-mls	0.3	—	3.329	3.052	3.445	3.688	3.41	3.054	3.328
qi-mls	0.3	0.1	3.032	3.011	3.327	3.065	3.267	2.285	3.239
qi-mls	0.3	0.2	3.05	3.014	3.342	3.082	3.277	2.34	3.246
qi-mls	0.3	0.3	3.062	3.017	3.345	3.102	3.291	2.362	3.253
qi-mls	0.3	0.4	3.072	3.026	3.35	3.114	3.296	2.381	3.267
qi-mls	0.3	0.5	3.083	3.03	3.354	3.146	3.3	2.395	3.272
qi-mls	0.3	0.6	3.093	3.032	3.359	3.163	3.309	2.415	3.28
qi-mls	0.3	0.7	3.104	3.037	3.364	3.179	3.311	2.44	3.29
qi-mls	0.3	0.8	3.118	3.041	3.373	3.203	3.324	2.467	3.299
qi-mls	0.3	0.9	3.147	3.053	3.384	3.25	3.34	2.537	3.316
qi-mls	0.5	—	10.528	10.186	10.79	10.16	10.422	11.145	10.466
qi-mls	0.5	0.1	10.272	10.15	10.728	9.566	10.32	10.496	10.373
qi-mls	0.5	0.2	10.282	10.151	10.736	9.578	10.324	10.52	10.38
qi-mls	0.5	0.3	10.287	10.151	10.742	9.587	10.326	10.535	10.384
qi-mls	0.5	0.4	10.292	10.151	10.745	9.596	10.327	10.546	10.387
qi-mls	0.5	0.5	10.296	10.151	10.747	9.604	10.328	10.555	10.389
qi-mls	0.5	0.6	10.299	10.151	10.75	9.613	10.328	10.564	10.39
qi-mls	0.5	0.7	10.303	10.152	10.751	9.624	10.329	10.574	10.391
qi-mls	0.5	0.8	10.309	10.152	10.754	9.638	10.33	10.588	10.392
qi-mls	0.5	0.9	10.322	10.153	10.759	9.667	10.334	10.625	10.396
qi-mls	0.7	—	10.528	10.186	10.79	10.16	10.422	11.145	10.466
qi-mls	0.7	0.1	10.272	10.15	10.728	9.566	10.32	10.496	10.373
qi-mls	0.7	0.2	10.282	10.151	10.736	9.578	10.324	10.52	10.38
qi-mls	0.7	0.3	10.287	10.151	10.742	9.587	10.326	10.535	10.384
qi-mls	0.7	0.4	10.292	10.151	10.745	9.596	10.327	10.546	10.387
qi-mls	0.7	0.5	10.296	10.151	10.747	9.604	10.328	10.555	10.389
qi-mls	0.7	0.6	10.299	10.151	10.75	9.613	10.328	10.564	10.39
qi-mls	0.7	0.7	10.303	10.152	10.751	9.624	10.329	10.574	10.391
qi-mls	0.7	0.8	10.309	10.152	10.754	9.638	10.33	10.588	10.392
qi-mls	0.7	0.9	10.322	10.153	10.759	9.667	10.334	10.625	10.396
qi-mls	0.9	—	10.528	10.186	10.79	10.16	10.422	11.145	10.466
qi-mls	0.9	0.1	10.272	10.15	10.728	9.566	10.32	10.496	10.373
qi-mls	0.9	0.2	10.282	10.151	10.736	9.578	10.324	10.52	10.38
qi-mls	0.9	0.3	10.287	10.151	10.742	9.587	10.326	10.535	10.384
qi-mls	0.9	0.4	10.292	10.151	10.745	9.596	10.327	10.546	10.387
qi-mls	0.9	0.5	10.296	10.151	10.747	9.604	10.328	10.555	10.389
qi-mls	0.9	0.6	10.299	10.151	10.75	9.613	10.328	10.564	10.39
qi-mls	0.9	0.7	10.303	10.152	10.751	9.624	10.329	10.574	10.391
qi-mls	0.9	0.8	10.309	10.152	10.754	9.638	10.33	10.588	10.392
qi-mls	0.9	0.9	10.322	10.153	10.759	9.667	10.334	10.625	10.396
qi-mls-ng	0.1	—	0.089	0.0	0.0	0.0	0.194	0.064	0.278
qi-mls-ng	0.1	0.1	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.1	0.2	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.1	0.3	0.078	0.0	0.0	0.0	0.148	0.054	0.267
qi-mls-ng	0.1	0.4	0.085	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.1	0.5	0.086	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.1	0.6	0.086	0.0	0.0	0.0	0.192	0.055	0.267
qi-mls-ng	0.1	0.7	0.087	0.0	0.0	0.0	0.192	0.055	0.274
qi-mls-ng	0.1	0.8	0.087	0.0	0.0	0.0	0.192	0.056	0.274
qi-mls-ng	0.1	0.9	0.087	0.0	0.0	0.0	0.192	0.057	0.274
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 15 (VOI_s on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-ng	0.3	—	0.089	0.0	0.0	0.0	0.194	0.064	0.278
qi-mls-ng	0.3	0.1	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.3	0.2	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.3	0.3	0.078	0.0	0.0	0.0	0.148	0.054	0.267
qi-mls-ng	0.3	0.4	0.085	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.3	0.5	0.086	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.3	0.6	0.086	0.0	0.0	0.0	0.192	0.055	0.267
qi-mls-ng	0.3	0.7	0.087	0.0	0.0	0.0	0.192	0.055	0.274
qi-mls-ng	0.3	0.8	0.087	0.0	0.0	0.0	0.192	0.056	0.274
qi-mls-ng	0.3	0.9	0.087	0.0	0.0	0.0	0.192	0.057	0.274
qi-mls-ng	0.5	—	0.089	0.0	0.0	0.0	0.194	0.064	0.278
qi-mls-ng	0.5	0.1	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.5	0.2	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.5	0.3	0.078	0.0	0.0	0.0	0.148	0.054	0.267
qi-mls-ng	0.5	0.4	0.085	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.5	0.5	0.086	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.5	0.6	0.086	0.0	0.0	0.0	0.192	0.055	0.267
qi-mls-ng	0.5	0.7	0.087	0.0	0.0	0.0	0.192	0.055	0.274
qi-mls-ng	0.5	0.8	0.087	0.0	0.0	0.0	0.192	0.056	0.274
qi-mls-ng	0.5	0.9	0.087	0.0	0.0	0.0	0.192	0.057	0.274
qi-mls-ng	0.7	—	0.089	0.0	0.0	0.0	0.194	0.064	0.278
qi-mls-ng	0.7	0.1	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.7	0.2	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.7	0.3	0.078	0.0	0.0	0.0	0.148	0.054	0.267
qi-mls-ng	0.7	0.4	0.085	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.7	0.5	0.086	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.7	0.6	0.086	0.0	0.0	0.0	0.192	0.055	0.267
qi-mls-ng	0.7	0.7	0.087	0.0	0.0	0.0	0.192	0.055	0.274
qi-mls-ng	0.7	0.8	0.087	0.0	0.0	0.0	0.192	0.056	0.274
qi-mls-ng	0.7	0.9	0.087	0.0	0.0	0.0	0.192	0.057	0.274
qi-mls-ng	0.9	—	0.089	0.0	0.0	0.0	0.194	0.064	0.278
qi-mls-ng	0.9	0.1	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.9	0.2	0.078	0.0	0.0	0.0	0.148	0.053	0.266
qi-mls-ng	0.9	0.3	0.078	0.0	0.0	0.0	0.148	0.054	0.267
qi-mls-ng	0.9	0.4	0.085	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.9	0.5	0.086	0.0	0.0	0.0	0.192	0.054	0.267
qi-mls-ng	0.9	0.6	0.086	0.0	0.0	0.0	0.192	0.055	0.267
qi-mls-ng	0.9	0.7	0.087	0.0	0.0	0.0	0.192	0.055	0.274
qi-mls-ng	0.9	0.8	0.087	0.0	0.0	0.0	0.192	0.056	0.274
qi-mls-ng	0.9	0.9	0.087	0.0	0.0	0.0	0.192	0.057	0.274
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Table 16: VOI_m performance on 25% of the data that were not used for training for all architectures and parameter sets. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging.

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	—	5.496	6.383	5.803	4.283	6.029	4.556	5.92
qi-mse	0.1	0.1	5.199	6.246	5.714	3.774	5.89	3.83	5.741
qi-mse	0.1	0.2	5.196	6.246	5.715	3.767	5.889	3.823	5.737
qi-mse	0.1	0.3	5.24	6.246	5.715	3.767	5.889	4.088	5.736
qi-mse	0.1	0.4	5.24	6.246	5.715	3.767	5.89	4.089	5.736
qi-mse	0.1	0.5	5.242	6.247	5.715	3.77	5.891	4.091	5.736
qi-mse	0.1	0.6	5.243	6.248	5.715	3.773	5.892	4.094	5.736
qi-mse	0.1	0.7	5.245	6.249	5.716	3.778	5.894	4.099	5.737
qi-mse	0.1	0.8	5.248	6.251	5.717	3.782	5.896	4.104	5.738
qi-mse	0.1	0.9	5.253	6.254	5.718	3.794	5.899	4.115	5.74
qi-mse	0.3	—	5.345	6.334	5.717	4.112	5.946	4.104	5.859
qi-mse	0.3	0.1	5.146	6.242	5.66	3.643	5.789	3.824	5.72
qi-mse	0.3	0.2	5.162	6.242	5.658	3.749	5.787	3.818	5.716
qi-mse	0.3	0.3	5.14	6.242	5.657	3.632	5.786	3.814	5.712
qi-mse	0.3	0.4	5.14	6.242	5.656	3.632	5.786	3.813	5.71
qi-mse	0.3	0.5	5.14	6.243	5.655	3.634	5.786	3.814	5.708
qi-mse	0.3	0.6	5.141	6.244	5.653	3.637	5.787	3.815	5.708
qi-mse	0.3	0.7	5.14	6.244	5.652	3.638	5.788	3.812	5.708
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 16 (VOI_m on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.3	0.8	5.14	6.245	5.651	3.638	5.789	3.811	5.708
qi-mse	0.3	0.9	5.139	6.246	5.651	3.644	5.781	3.803	5.709
qi-mse	0.5	—	1.261	1.192	2.112	0.54	0.892	1.86	0.968
qi-mse	0.5	0.1	1.235	1.135	2.708	0.447	1.021	1.08	1.017
qi-mse	0.5	0.2	1.203	1.082	2.673	0.427	1.01	1.045	0.983
qi-mse	0.5	0.3	1.207	1.096	2.625	0.419	0.95	1.265	0.887
qi-mse	0.5	0.4	1.22	1.08	2.583	0.415	0.974	1.254	1.013
qi-mse	0.5	0.5	1.156	1.161	2.575	0.413	0.894	1.018	0.879
qi-mse	0.5	0.6	1.079	1.088	2.16	0.41	0.868	1.064	0.882
qi-mse	0.5	0.7	1.069	1.018	2.158	0.402	0.866	1.059	0.91
qi-mse	0.5	0.8	1.08	0.976	2.157	0.402	0.863	1.173	0.909
qi-mse	0.5	0.9	1.042	1.062	2.152	0.403	0.737	0.996	0.906
qi-mse	0.7	—	0.198	0.05	0.412	0.206	0.082	0.259	0.176
qi-mse	0.7	0.1	0.187	0.024	0.445	0.149	0.192	0.163	0.15
qi-mse	0.7	0.2	0.162	0.017	0.425	0.129	0.14	0.13	0.134
qi-mse	0.7	0.3	0.154	0.015	0.416	0.12	0.133	0.114	0.125
qi-mse	0.7	0.4	0.149	0.014	0.41	0.116	0.128	0.105	0.12
qi-mse	0.7	0.5	0.146	0.014	0.406	0.114	0.126	0.099	0.117
qi-mse	0.7	0.6	0.143	0.014	0.401	0.112	0.123	0.094	0.116
qi-mse	0.7	0.7	0.141	0.014	0.398	0.111	0.119	0.09	0.114
qi-mse	0.7	0.8	0.139	0.014	0.396	0.111	0.116	0.086	0.113
qi-mse	0.7	0.9	0.138	0.014	0.395	0.112	0.113	0.084	0.112
qi-mse	0.9	—	0.115	0.032	0.187	0.109	0.064	0.187	0.113
qi-mse	0.9	0.1	0.117	0.02	0.22	0.076	0.145	0.127	0.111
qi-mse	0.9	0.2	0.098	0.014	0.2	0.056	0.132	0.094	0.093
qi-mse	0.9	0.3	0.09	0.012	0.19	0.047	0.125	0.078	0.085
qi-mse	0.9	0.4	0.085	0.011	0.184	0.043	0.121	0.069	0.081
qi-mse	0.9	0.5	0.081	0.011	0.18	0.04	0.118	0.062	0.078
qi-mse	0.9	0.6	0.079	0.011	0.177	0.039	0.115	0.057	0.076
qi-mse	0.9	0.7	0.077	0.011	0.174	0.037	0.113	0.053	0.074
qi-mse	0.9	0.8	0.075	0.011	0.172	0.037	0.11	0.049	0.073
qi-mse	0.9	0.9	0.074	0.011	0.171	0.038	0.107	0.048	0.072
qi-mse-ng	0.1	—	5.505	6.382	5.804	4.286	6.038	4.592	5.931
qi-mse-ng	0.1	0.1	5.229	6.244	5.714	3.738	5.835	4.106	5.74
qi-mse-ng	0.1	0.2	5.227	6.243	5.713	3.735	5.833	4.102	5.736
qi-mse-ng	0.1	0.3	5.226	6.242	5.712	3.734	5.832	4.102	5.734
qi-mse-ng	0.1	0.4	5.227	6.242	5.712	3.736	5.831	4.103	5.735
qi-mse-ng	0.1	0.5	5.238	6.243	5.712	3.805	5.831	4.105	5.734
qi-mse-ng	0.1	0.6	5.24	6.243	5.714	3.808	5.831	4.109	5.734
qi-mse-ng	0.1	0.7	5.242	6.243	5.714	3.814	5.831	4.117	5.734
qi-mse-ng	0.1	0.8	5.245	6.244	5.715	3.821	5.832	4.125	5.734
qi-mse-ng	0.1	0.9	5.239	6.245	5.716	3.767	5.832	4.139	5.734
qi-mse-ng	0.3	—	5.427	6.35	5.768	4.115	5.964	4.454	5.91
qi-mse-ng	0.3	0.1	5.226	6.242	5.707	3.735	5.832	4.104	5.737
qi-mse-ng	0.3	0.2	5.222	6.241	5.698	3.732	5.83	4.099	5.732
qi-mse-ng	0.3	0.3	5.221	6.241	5.698	3.731	5.829	4.098	5.731
qi-mse-ng	0.3	0.4	5.221	6.241	5.697	3.731	5.828	4.098	5.729
qi-mse-ng	0.3	0.5	5.221	6.241	5.699	3.732	5.825	4.1	5.726
qi-mse-ng	0.3	0.6	5.221	6.241	5.699	3.734	5.825	4.103	5.726
qi-mse-ng	0.3	0.7	5.221	6.242	5.696	3.733	5.825	4.106	5.725
qi-mse-ng	0.3	0.8	5.161	6.242	5.697	3.366	5.825	4.109	5.725
qi-mse-ng	0.3	0.9	5.163	6.243	5.698	3.37	5.826	4.118	5.725
qi-mse-ng	0.5	—	3.563	3.186	4.414	1.378	4.782	3.603	4.014
qi-mse-ng	0.5	0.1	3.547	3.151	4.528	1.319	4.651	3.327	4.306
qi-mse-ng	0.5	0.2	3.506	2.962	4.52	1.281	4.722	3.297	4.253
qi-mse-ng	0.5	0.3	3.544	3.039	4.515	1.427	4.72	3.286	4.278
qi-mse-ng	0.5	0.4	3.516	3.045	4.508	1.298	4.719	3.285	4.24
qi-mse-ng	0.5	0.5	3.516	3.155	4.503	1.262	4.618	3.322	4.238
qi-mse-ng	0.5	0.6	3.498	3.107	4.499	1.258	4.617	3.276	4.232
qi-mse-ng	0.5	0.7	3.436	2.94	4.488	1.144	4.549	3.267	4.229
qi-mse-ng	0.5	0.8	3.463	3.003	4.487	1.268	4.548	3.257	4.212
qi-mse-ng	0.5	0.9	3.464	3.028	4.467	1.268	4.549	3.257	4.213
qi-mse-ng	0.7	—	0.281	0.161	0.475	0.216	0.181	0.37	0.283
qi-mse-ng	0.7	0.1	0.226	0.107	0.504	0.177	0.128	0.231	0.207
qi-mse-ng	0.7	0.2	0.212	0.104	0.49	0.158	0.12	0.208	0.193
qi-mse-ng	0.7	0.3	0.206	0.103	0.483	0.149	0.116	0.197	0.187
qi-mse-ng	0.7	0.4	0.2	0.102	0.469	0.143	0.114	0.189	0.182
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 16 (VOI_m on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse-ng	0.7	0.5	0.197	0.102	0.465	0.139	0.112	0.184	0.179
qi-mse-ng	0.7	0.6	0.195	0.102	0.463	0.136	0.111	0.179	0.177
qi-mse-ng	0.7	0.7	0.193	0.102	0.461	0.133	0.111	0.176	0.176
qi-mse-ng	0.7	0.8	0.192	0.102	0.459	0.131	0.111	0.172	0.175
qi-mse-ng	0.7	0.9	0.192	0.102	0.458	0.131	0.111	0.173	0.175
qi-mse-ng	0.9	—	0.13	0.051	0.215	0.105	0.073	0.204	0.129
qi-mse-ng	0.9	0.1	0.095	0.02	0.227	0.082	0.031	0.114	0.096
qi-mse-ng	0.9	0.2	0.081	0.017	0.213	0.063	0.022	0.091	0.082
qi-mse-ng	0.9	0.3	0.075	0.016	0.207	0.055	0.019	0.079	0.076
qi-mse-ng	0.9	0.4	0.071	0.016	0.202	0.049	0.016	0.07	0.071
qi-mse-ng	0.9	0.5	0.068	0.015	0.198	0.045	0.015	0.064	0.068
qi-mse-ng	0.9	0.6	0.065	0.015	0.196	0.041	0.014	0.059	0.066
qi-mse-ng	0.9	0.7	0.063	0.015	0.193	0.038	0.013	0.055	0.065
qi-mse-ng	0.9	0.8	0.062	0.015	0.192	0.036	0.013	0.053	0.064
qi-mse-ng	0.9	0.9	0.062	0.015	0.192	0.036	0.014	0.053	0.064
qi-mls-pre	0.1	—	3.01	2.551	3.506	1.903	4.037	2.58	3.481
qi-mls-pre	0.1	0.1	2.967	2.722	3.642	1.555	3.892	2.42	3.572
qi-mls-pre	0.1	0.2	2.945	2.723	3.622	1.532	3.821	2.397	3.575
qi-mls-pre	0.1	0.3	2.927	2.722	3.647	1.522	3.809	2.27	3.59
qi-mls-pre	0.1	0.4	2.938	2.722	3.644	1.477	3.807	2.375	3.602
qi-mls-pre	0.1	0.5	2.872	2.722	3.641	1.473	3.789	2.011	3.592
qi-mls-pre	0.1	0.6	2.873	2.754	3.639	1.471	3.788	1.997	3.591
qi-mls-pre	0.1	0.7	2.871	2.743	3.638	1.472	3.787	1.996	3.589
qi-mls-pre	0.1	0.8	2.856	2.744	3.549	1.473	3.787	1.996	3.588
qi-mls-pre	0.1	0.9	2.855	2.745	3.544	1.478	3.779	1.994	3.588
qi-mls-pre	0.3	—	1.061	1.121	1.2	1.02	1.103	0.648	1.272
qi-mls-pre	0.3	0.1	1.013	1.071	1.22	0.891	1.062	0.608	1.227
qi-mls-pre	0.3	0.2	0.996	1.068	1.199	0.867	1.05	0.578	1.213
qi-mls-pre	0.3	0.3	0.989	1.067	1.188	0.855	1.044	0.557	1.226
qi-mls-pre	0.3	0.4	0.983	1.067	1.179	0.848	1.041	0.545	1.221
qi-mls-pre	0.3	0.5	0.981	1.067	1.173	0.845	1.039	0.543	1.218
qi-mls-pre	0.3	0.6	0.977	1.067	1.168	0.842	1.037	0.534	1.216
qi-mls-pre	0.3	0.7	0.975	1.067	1.164	0.842	1.035	0.529	1.214
qi-mls-pre	0.3	0.8	0.974	1.067	1.161	0.843	1.035	0.526	1.213
qi-mls-pre	0.3	0.9	0.973	1.068	1.158	0.846	1.035	0.52	1.211
qi-mls-pre	0.5	—	1.052	1.121	1.193	1.006	1.103	0.63	1.262
qi-mls-pre	0.5	0.1	1.006	1.071	1.213	0.88	1.062	0.606	1.205
qi-mls-pre	0.5	0.2	0.989	1.068	1.193	0.856	1.05	0.575	1.191
qi-mls-pre	0.5	0.3	0.982	1.067	1.181	0.845	1.044	0.554	1.204
qi-mls-pre	0.5	0.4	0.976	1.067	1.172	0.838	1.041	0.542	1.199
qi-mls-pre	0.5	0.5	0.973	1.067	1.166	0.834	1.039	0.54	1.196
qi-mls-pre	0.5	0.6	0.97	1.067	1.161	0.832	1.037	0.531	1.194
qi-mls-pre	0.5	0.7	0.968	1.067	1.157	0.832	1.035	0.527	1.192
qi-mls-pre	0.5	0.8	0.967	1.067	1.154	0.832	1.035	0.523	1.191
qi-mls-pre	0.5	0.9	0.966	1.068	1.151	0.835	1.035	0.517	1.189
qi-mls-pre	0.7	—	1.052	1.121	1.193	1.006	1.103	0.63	1.262
qi-mls-pre	0.7	0.1	1.006	1.071	1.213	0.88	1.062	0.606	1.205
qi-mls-pre	0.7	0.2	0.989	1.068	1.193	0.856	1.05	0.575	1.191
qi-mls-pre	0.7	0.3	0.982	1.067	1.181	0.845	1.044	0.554	1.204
qi-mls-pre	0.7	0.4	0.976	1.067	1.172	0.838	1.041	0.542	1.199
qi-mls-pre	0.7	0.5	0.973	1.067	1.166	0.834	1.039	0.54	1.196
qi-mls-pre	0.7	0.6	0.97	1.067	1.161	0.832	1.037	0.531	1.194
qi-mls-pre	0.7	0.7	0.968	1.067	1.157	0.832	1.035	0.527	1.192
qi-mls-pre	0.7	0.8	0.967	1.067	1.154	0.832	1.035	0.523	1.191
qi-mls-pre	0.7	0.9	0.966	1.068	1.151	0.835	1.035	0.517	1.189
qi-mls-pre	0.9	—	1.052	1.121	1.193	1.006	1.103	0.63	1.262
qi-mls-pre	0.9	0.1	1.006	1.071	1.213	0.88	1.062	0.606	1.205
qi-mls-pre	0.9	0.2	0.989	1.068	1.193	0.856	1.05	0.575	1.191
qi-mls-pre	0.9	0.3	0.982	1.067	1.181	0.845	1.044	0.554	1.204
qi-mls-pre	0.9	0.4	0.976	1.067	1.172	0.838	1.041	0.542	1.199
qi-mls-pre	0.9	0.5	0.973	1.067	1.166	0.834	1.039	0.54	1.196
qi-mls-pre	0.9	0.6	0.97	1.067	1.161	0.832	1.037	0.531	1.194
qi-mls-pre	0.9	0.7	0.968	1.067	1.157	0.832	1.035	0.527	1.192
qi-mls-pre	0.9	0.8	0.967	1.067	1.154	0.832	1.035	0.523	1.191
qi-mls-pre	0.9	0.9	0.966	1.068	1.151	0.835	1.035	0.517	1.189
qi-mls-pre-ng	0.1	—	4.668	4.913	4.89	3.58	5.161	4.435	5.028
qi-mls-pre-ng	0.1	0.1	4.525	5.103	4.921	3.185	4.867	3.955	5.119
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 16 (VOI_m on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre-ng	0.1	0.2	4.513	5.158	4.957	3.093	4.825	3.966	5.081
qi-mls-pre-ng	0.1	0.3	4.484	5.124	4.842	3.088	4.822	3.969	5.057
qi-mls-pre-ng	0.1	0.4	4.475	5.08	4.839	3.087	4.821	3.97	5.056
qi-mls-pre-ng	0.1	0.5	4.477	5.082	4.843	3.088	4.82	3.974	5.055
qi-mls-pre-ng	0.1	0.6	4.478	5.081	4.844	3.091	4.82	3.978	5.054
qi-mls-pre-ng	0.1	0.7	4.477	5.08	4.837	3.094	4.82	3.981	5.054
qi-mls-pre-ng	0.1	0.8	4.464	5.079	4.832	3.072	4.819	3.928	5.053
qi-mls-pre-ng	0.1	0.9	4.464	5.079	4.835	3.111	4.757	4.012	4.988
qi-mls-pre-ng	0.3	—	2.275	2.431	2.055	2.112	2.351	1.874	2.829
qi-mls-pre-ng	0.3	0.1	2.197	2.372	2.075	1.94	2.193	1.871	2.729
qi-mls-pre-ng	0.3	0.2	2.17	2.377	2.048	1.918	2.18	1.777	2.72
qi-mls-pre-ng	0.3	0.3	2.169	2.376	2.037	1.936	2.176	1.77	2.717
qi-mls-pre-ng	0.3	0.4	2.162	2.385	2.031	1.936	2.175	1.731	2.714
qi-mls-pre-ng	0.3	0.5	2.163	2.385	2.038	1.936	2.174	1.73	2.713
qi-mls-pre-ng	0.3	0.6	2.161	2.385	2.036	1.938	2.173	1.72	2.713
qi-mls-pre-ng	0.3	0.7	2.16	2.385	2.035	1.94	2.173	1.716	2.712
qi-mls-pre-ng	0.3	0.8	2.16	2.386	2.033	1.936	2.173	1.718	2.711
qi-mls-pre-ng	0.3	0.9	2.148	2.39	2.035	1.952	2.174	1.624	2.711
qi-mls-pre-ng	0.5	—	2.275	2.431	2.055	2.112	2.351	1.874	2.829
qi-mls-pre-ng	0.5	0.1	2.197	2.372	2.075	1.94	2.193	1.871	2.729
qi-mls-pre-ng	0.5	0.2	2.17	2.377	2.048	1.918	2.18	1.777	2.72
qi-mls-pre-ng	0.5	0.3	2.169	2.376	2.037	1.936	2.176	1.77	2.717
qi-mls-pre-ng	0.5	0.4	2.162	2.385	2.031	1.936	2.175	1.731	2.714
qi-mls-pre-ng	0.5	0.5	2.163	2.385	2.038	1.936	2.174	1.73	2.713
qi-mls-pre-ng	0.5	0.6	2.161	2.385	2.036	1.938	2.173	1.72	2.713
qi-mls-pre-ng	0.5	0.7	2.16	2.385	2.035	1.94	2.173	1.716	2.712
qi-mls-pre-ng	0.5	0.8	2.16	2.386	2.033	1.936	2.173	1.718	2.711
qi-mls-pre-ng	0.5	0.9	2.148	2.39	2.035	1.952	2.174	1.624	2.711
qi-mls-pre-ng	0.7	—	2.275	2.431	2.055	2.112	2.351	1.874	2.829
qi-mls-pre-ng	0.7	0.1	2.197	2.372	2.075	1.94	2.193	1.871	2.729
qi-mls-pre-ng	0.7	0.2	2.17	2.377	2.048	1.918	2.18	1.777	2.72
qi-mls-pre-ng	0.7	0.3	2.169	2.376	2.037	1.936	2.176	1.77	2.717
qi-mls-pre-ng	0.7	0.4	2.162	2.385	2.031	1.936	2.175	1.731	2.714
qi-mls-pre-ng	0.7	0.5	2.163	2.385	2.038	1.936	2.174	1.73	2.713
qi-mls-pre-ng	0.7	0.6	2.161	2.385	2.036	1.938	2.173	1.72	2.713
qi-mls-pre-ng	0.7	0.7	2.16	2.385	2.035	1.94	2.173	1.716	2.712
qi-mls-pre-ng	0.7	0.8	2.16	2.386	2.033	1.936	2.173	1.718	2.711
qi-mls-pre-ng	0.7	0.9	2.148	2.39	2.035	1.952	2.174	1.624	2.711
qi-mls-pre-ng	0.9	—	2.275	2.431	2.055	2.112	2.351	1.874	2.829
qi-mls-pre-ng	0.9	0.1	2.197	2.372	2.075	1.94	2.193	1.871	2.729
qi-mls-pre-ng	0.9	0.2	2.17	2.377	2.048	1.918	2.18	1.777	2.72
qi-mls-pre-ng	0.9	0.3	2.169	2.376	2.037	1.936	2.176	1.77	2.717
qi-mls-pre-ng	0.9	0.4	2.162	2.385	2.031	1.936	2.175	1.731	2.714
qi-mls-pre-ng	0.9	0.5	2.163	2.385	2.038	1.936	2.174	1.73	2.713
qi-mls-pre-ng	0.9	0.6	2.161	2.385	2.036	1.938	2.173	1.72	2.713
qi-mls-pre-ng	0.9	0.7	2.16	2.385	2.035	1.94	2.173	1.716	2.712
qi-mls-pre-ng	0.9	0.8	2.16	2.386	2.033	1.936	2.173	1.718	2.711
qi-mls-pre-ng	0.9	0.9	2.148	2.39	2.035	1.952	2.174	1.624	2.711
qi-mls	0.1	—	0.367	0.158	0.496	0.404	0.262	0.53	0.355
qi-mls	0.1	0.1	0.274	0.121	0.484	0.297	0.175	0.23	0.337
qi-mls	0.1	0.2	0.259	0.118	0.466	0.28	0.165	0.201	0.321
qi-mls	0.1	0.3	0.248	0.116	0.454	0.271	0.158	0.178	0.31
qi-mls	0.1	0.4	0.242	0.116	0.447	0.268	0.154	0.165	0.303
qi-mls	0.1	0.5	0.237	0.116	0.443	0.262	0.149	0.156	0.298
qi-mls	0.1	0.6	0.234	0.116	0.441	0.26	0.148	0.148	0.294
qi-mls	0.1	0.7	0.233	0.116	0.44	0.259	0.147	0.144	0.291
qi-mls	0.1	0.8	0.231	0.116	0.433	0.26	0.146	0.139	0.29
qi-mls	0.1	0.9	0.233	0.117	0.435	0.266	0.147	0.143	0.289
qi-mls	0.3	—	0.186	0.068	0.234	0.185	0.15	0.28	0.201
qi-mls	0.3	0.1	0.176	0.058	0.25	0.174	0.135	0.199	0.241
qi-mls	0.3	0.2	0.159	0.054	0.233	0.156	0.123	0.166	0.224
qi-mls	0.3	0.3	0.15	0.053	0.223	0.147	0.115	0.146	0.213
qi-mls	0.3	0.4	0.143	0.052	0.214	0.14	0.111	0.132	0.206
qi-mls	0.3	0.5	0.138	0.052	0.209	0.136	0.108	0.122	0.201
qi-mls	0.3	0.6	0.135	0.052	0.206	0.134	0.107	0.115	0.197
qi-mls	0.3	0.7	0.133	0.052	0.203	0.133	0.106	0.11	0.195
qi-mls	0.3	0.8	0.132	0.053	0.202	0.133	0.105	0.105	0.193
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 16 (VOI_m on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls	0.3	0.9	0.132	0.053	0.202	0.136	0.105	0.106	0.191
qi-mls	0.5	—	0.084	0.027	0.127	0.097	0.068	0.079	0.105
qi-mls	0.5	0.1	0.13	0.04	0.169	0.127	0.092	0.172	0.18
qi-mls	0.5	0.2	0.112	0.036	0.15	0.109	0.079	0.137	0.163
qi-mls	0.5	0.3	0.102	0.035	0.138	0.099	0.072	0.117	0.152
qi-mls	0.5	0.4	0.095	0.034	0.13	0.093	0.068	0.103	0.145
qi-mls	0.5	0.5	0.09	0.034	0.124	0.088	0.065	0.092	0.139
qi-mls	0.5	0.6	0.087	0.033	0.12	0.086	0.063	0.084	0.135
qi-mls	0.5	0.7	0.084	0.033	0.116	0.084	0.061	0.078	0.133
qi-mls	0.5	0.8	0.082	0.033	0.114	0.084	0.061	0.072	0.13
qi-mls	0.5	0.9	0.081	0.033	0.112	0.084	0.06	0.067	0.128
qi-mls	0.7	—	0.084	0.027	0.127	0.097	0.068	0.079	0.105
qi-mls	0.7	0.1	0.13	0.04	0.169	0.127	0.092	0.172	0.18
qi-mls	0.7	0.2	0.112	0.036	0.15	0.109	0.079	0.137	0.163
qi-mls	0.7	0.3	0.102	0.035	0.138	0.099	0.072	0.117	0.152
qi-mls	0.7	0.4	0.095	0.034	0.13	0.093	0.068	0.103	0.145
qi-mls	0.7	0.5	0.09	0.034	0.124	0.088	0.065	0.092	0.139
qi-mls	0.7	0.6	0.087	0.033	0.12	0.086	0.063	0.084	0.135
qi-mls	0.7	0.7	0.084	0.033	0.116	0.084	0.061	0.078	0.133
qi-mls	0.7	0.8	0.082	0.033	0.114	0.084	0.061	0.072	0.13
qi-mls	0.7	0.9	0.081	0.033	0.112	0.084	0.06	0.067	0.128
qi-mls	0.9	—	0.084	0.027	0.127	0.097	0.068	0.079	0.105
qi-mls	0.9	0.1	0.13	0.04	0.169	0.127	0.092	0.172	0.18
qi-mls	0.9	0.2	0.112	0.036	0.15	0.109	0.079	0.137	0.163
qi-mls	0.9	0.3	0.102	0.035	0.138	0.099	0.072	0.117	0.152
qi-mls	0.9	0.4	0.095	0.034	0.13	0.093	0.068	0.103	0.145
qi-mls	0.9	0.5	0.09	0.034	0.124	0.088	0.065	0.092	0.139
qi-mls	0.9	0.6	0.087	0.033	0.12	0.086	0.063	0.084	0.135
qi-mls	0.9	0.7	0.084	0.033	0.116	0.084	0.061	0.078	0.133
qi-mls	0.9	0.8	0.082	0.033	0.114	0.084	0.061	0.072	0.13
qi-mls	0.9	0.9	0.081	0.033	0.112	0.084	0.06	0.067	0.128
qi-mls-ng	0.1	—	5.56	6.385	5.804	5.07	5.773	4.633	5.695
qi-mls-ng	0.1	0.1	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.1	0.2	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.1	0.3	5.567	6.385	5.804	5.07	5.811	4.631	5.699
qi-mls-ng	0.1	0.4	5.562	6.385	5.804	5.07	5.785	4.631	5.699
qi-mls-ng	0.1	0.5	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.1	0.6	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.1	0.7	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.1	0.8	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.1	0.9	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.3	—	5.56	6.385	5.804	5.07	5.773	4.633	5.695
qi-mls-ng	0.3	0.1	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.3	0.2	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.3	0.3	5.567	6.385	5.804	5.07	5.811	4.631	5.699
qi-mls-ng	0.3	0.4	5.562	6.385	5.804	5.07	5.785	4.631	5.699
qi-mls-ng	0.3	0.5	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.3	0.6	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.3	0.7	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.3	0.8	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.3	0.9	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.5	—	5.56	6.385	5.804	5.07	5.773	4.633	5.695
qi-mls-ng	0.5	0.1	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.5	0.2	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.5	0.3	5.567	6.385	5.804	5.07	5.811	4.631	5.699
qi-mls-ng	0.5	0.4	5.562	6.385	5.804	5.07	5.785	4.631	5.699
qi-mls-ng	0.5	0.5	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.5	0.6	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.5	0.7	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.5	0.8	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.5	0.9	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.7	—	5.56	6.385	5.804	5.07	5.773	4.633	5.695
qi-mls-ng	0.7	0.1	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.7	0.2	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.7	0.3	5.567	6.385	5.804	5.07	5.811	4.631	5.699
qi-mls-ng	0.7	0.4	5.562	6.385	5.804	5.07	5.785	4.631	5.699
qi-mls-ng	0.7	0.5	5.562	6.385	5.804	5.07	5.785	4.632	5.699
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 16 (VOI_m on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-ng	0.7	0.6	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.7	0.7	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.7	0.8	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.7	0.9	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.9	—	5.56	6.385	5.804	5.07	5.773	4.633	5.695
qi-mls-ng	0.9	0.1	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.9	0.2	5.567	6.385	5.804	5.07	5.811	4.631	5.7
qi-mls-ng	0.9	0.3	5.567	6.385	5.804	5.07	5.811	4.631	5.699
qi-mls-ng	0.9	0.4	5.562	6.385	5.804	5.07	5.785	4.631	5.699
qi-mls-ng	0.9	0.5	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.9	0.6	5.562	6.385	5.804	5.07	5.785	4.632	5.699
qi-mls-ng	0.9	0.7	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.9	0.8	5.561	6.385	5.804	5.07	5.785	4.632	5.693
qi-mls-ng	0.9	0.9	5.561	6.385	5.804	5.07	5.785	4.632	5.693
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Table 17: RAND performance on 25% of the data that were not used for training for all architectures and parameter sets. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging.

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	—	0.917	0.969	0.938	0.804	0.961	0.891	0.942
qi-mse	0.1	0.1	0.919	0.968	0.936	0.826	0.961	0.882	0.939
qi-mse	0.1	0.2	0.919	0.968	0.936	0.827	0.961	0.882	0.94
qi-mse	0.1	0.3	0.921	0.968	0.936	0.827	0.961	0.894	0.94
qi-mse	0.1	0.4	0.921	0.968	0.936	0.827	0.961	0.894	0.94
qi-mse	0.1	0.5	0.921	0.968	0.936	0.828	0.961	0.895	0.94
qi-mse	0.1	0.6	0.921	0.968	0.936	0.828	0.961	0.895	0.94
qi-mse	0.1	0.7	0.921	0.968	0.936	0.828	0.961	0.895	0.94
qi-mse	0.1	0.8	0.921	0.968	0.936	0.828	0.961	0.895	0.94
qi-mse	0.1	0.9	0.922	0.968	0.937	0.829	0.961	0.896	0.94
qi-mse	0.3	—	0.92	0.969	0.937	0.826	0.961	0.888	0.942
qi-mse	0.3	0.1	0.918	0.968	0.936	0.822	0.96	0.881	0.939
qi-mse	0.3	0.2	0.919	0.968	0.936	0.826	0.96	0.882	0.939
qi-mse	0.3	0.3	0.918	0.968	0.936	0.823	0.96	0.882	0.939
qi-mse	0.3	0.4	0.918	0.968	0.936	0.823	0.96	0.882	0.939
qi-mse	0.3	0.5	0.918	0.968	0.936	0.823	0.96	0.883	0.939
qi-mse	0.3	0.6	0.918	0.968	0.936	0.823	0.96	0.883	0.939
qi-mse	0.3	0.7	0.918	0.968	0.936	0.823	0.96	0.883	0.939
qi-mse	0.3	0.8	0.918	0.968	0.936	0.823	0.96	0.883	0.939
qi-mse	0.3	0.9	0.918	0.968	0.936	0.824	0.96	0.883	0.939
qi-mse	0.5	—	0.55	0.542	0.763	0.53	0.428	0.677	0.359
qi-mse	0.5	0.1	0.528	0.513	0.841	0.53	0.454	0.471	0.358
qi-mse	0.5	0.2	0.523	0.502	0.839	0.529	0.453	0.469	0.347
qi-mse	0.5	0.3	0.53	0.518	0.837	0.529	0.443	0.537	0.315
qi-mse	0.5	0.4	0.535	0.502	0.835	0.529	0.449	0.536	0.358
qi-mse	0.5	0.5	0.521	0.553	0.835	0.528	0.421	0.469	0.32
qi-mse	0.5	0.6	0.505	0.517	0.768	0.529	0.418	0.478	0.321
qi-mse	0.5	0.7	0.503	0.494	0.768	0.529	0.418	0.478	0.329
qi-mse	0.5	0.8	0.51	0.486	0.768	0.529	0.418	0.529	0.329
qi-mse	0.5	0.9	0.488	0.51	0.768	0.529	0.326	0.468	0.329
qi-mse	0.7	—	0.224	0.062	0.286	0.515	0.115	0.26	0.106
qi-mse	0.7	0.1	0.221	0.052	0.287	0.516	0.151	0.231	0.092
qi-mse	0.7	0.2	0.217	0.051	0.286	0.516	0.129	0.229	0.091
qi-mse	0.7	0.3	0.217	0.052	0.285	0.516	0.128	0.228	0.092
qi-mse	0.7	0.4	0.217	0.052	0.285	0.517	0.128	0.228	0.092
qi-mse	0.7	0.5	0.217	0.052	0.285	0.517	0.127	0.228	0.092
qi-mse	0.7	0.6	0.217	0.052	0.285	0.517	0.127	0.228	0.092
qi-mse	0.7	0.7	0.217	0.052	0.285	0.517	0.128	0.228	0.092
qi-mse	0.7	0.8	0.217	0.052	0.285	0.517	0.129	0.228	0.092
qi-mse	0.7	0.9	0.218	0.052	0.285	0.518	0.129	0.229	0.092
qi-mse	0.9	—	0.39	0.517	0.302	0.556	0.408	0.323	0.234
qi-mse	0.9	0.1	0.385	0.503	0.303	0.556	0.408	0.297	0.242
qi-mse	0.9	0.2	0.384	0.503	0.302	0.555	0.407	0.296	0.242
qi-mse	0.9	0.3	0.385	0.503	0.302	0.555	0.411	0.296	0.242
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 17 (RAND on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.9	0.4	0.385	0.503	0.302	0.555	0.411	0.295	0.242
qi-mse	0.9	0.5	0.385	0.504	0.302	0.555	0.41	0.295	0.242
qi-mse	0.9	0.6	0.387	0.504	0.302	0.555	0.411	0.304	0.244
qi-mse	0.9	0.7	0.387	0.504	0.302	0.556	0.41	0.305	0.244
qi-mse	0.9	0.8	0.387	0.504	0.302	0.556	0.411	0.305	0.244
qi-mse	0.9	0.9	0.388	0.504	0.302	0.557	0.414	0.306	0.245
qi-mse-ng	0.1	—	0.917	0.969	0.938	0.803	0.961	0.888	0.942
qi-mse-ng	0.1	0.1	0.92	0.968	0.936	0.824	0.96	0.894	0.94
qi-mse-ng	0.1	0.2	0.921	0.968	0.936	0.824	0.96	0.894	0.94
qi-mse-ng	0.1	0.3	0.921	0.968	0.936	0.825	0.96	0.895	0.94
qi-mse-ng	0.1	0.4	0.921	0.968	0.936	0.825	0.96	0.895	0.94
qi-mse-ng	0.1	0.5	0.921	0.968	0.936	0.829	0.96	0.895	0.94
qi-mse-ng	0.1	0.6	0.922	0.968	0.936	0.829	0.96	0.895	0.94
qi-mse-ng	0.1	0.7	0.922	0.968	0.936	0.83	0.96	0.896	0.94
qi-mse-ng	0.1	0.8	0.922	0.968	0.936	0.83	0.96	0.896	0.94
qi-mse-ng	0.1	0.9	0.921	0.968	0.937	0.827	0.96	0.896	0.94
qi-mse-ng	0.3	—	0.922	0.969	0.937	0.828	0.961	0.896	0.942
qi-mse-ng	0.3	0.1	0.92	0.968	0.936	0.824	0.96	0.894	0.94
qi-mse-ng	0.3	0.2	0.92	0.968	0.936	0.824	0.96	0.894	0.94
qi-mse-ng	0.3	0.3	0.921	0.968	0.936	0.824	0.96	0.895	0.94
qi-mse-ng	0.3	0.4	0.921	0.968	0.936	0.825	0.96	0.895	0.94
qi-mse-ng	0.3	0.5	0.921	0.968	0.936	0.825	0.96	0.895	0.94
qi-mse-ng	0.3	0.6	0.921	0.968	0.936	0.825	0.96	0.895	0.94
qi-mse-ng	0.3	0.7	0.921	0.968	0.936	0.825	0.96	0.895	0.94
qi-mse-ng	0.3	0.8	0.913	0.968	0.936	0.778	0.96	0.896	0.94
qi-mse-ng	0.3	0.9	0.913	0.968	0.936	0.778	0.96	0.896	0.94
qi-mse-ng	0.5	—	0.853	0.851	0.914	0.627	0.947	0.876	0.905
qi-mse-ng	0.5	0.1	0.857	0.856	0.915	0.643	0.944	0.868	0.917
qi-mse-ng	0.5	0.2	0.854	0.842	0.915	0.639	0.946	0.868	0.916
qi-mse-ng	0.5	0.3	0.857	0.849	0.915	0.649	0.946	0.868	0.917
qi-mse-ng	0.5	0.4	0.857	0.853	0.916	0.643	0.946	0.868	0.916
qi-mse-ng	0.5	0.5	0.857	0.859	0.915	0.638	0.944	0.87	0.916
qi-mse-ng	0.5	0.6	0.856	0.856	0.915	0.638	0.944	0.868	0.916
qi-mse-ng	0.5	0.7	0.851	0.844	0.915	0.618	0.943	0.868	0.916
qi-mse-ng	0.5	0.8	0.853	0.849	0.915	0.629	0.943	0.868	0.916
qi-mse-ng	0.5	0.9	0.853	0.849	0.915	0.629	0.943	0.868	0.916
qi-mse-ng	0.7	—	0.245	0.091	0.303	0.517	0.155	0.269	0.136
qi-mse-ng	0.7	0.1	0.231	0.085	0.303	0.521	0.129	0.244	0.106
qi-mse-ng	0.7	0.2	0.233	0.085	0.303	0.52	0.143	0.243	0.105
qi-mse-ng	0.7	0.3	0.23	0.085	0.302	0.52	0.127	0.242	0.105
qi-mse-ng	0.7	0.4	0.233	0.085	0.302	0.519	0.144	0.242	0.104
qi-mse-ng	0.7	0.5	0.233	0.085	0.302	0.519	0.144	0.242	0.104
qi-mse-ng	0.7	0.6	0.233	0.085	0.302	0.519	0.144	0.242	0.104
qi-mse-ng	0.7	0.7	0.233	0.085	0.302	0.519	0.144	0.243	0.104
qi-mse-ng	0.7	0.8	0.233	0.085	0.302	0.519	0.144	0.243	0.105
qi-mse-ng	0.7	0.9	0.233	0.086	0.302	0.519	0.144	0.244	0.105
qi-mse-ng	0.9	—	0.358	0.461	0.279	0.549	0.341	0.308	0.213
qi-mse-ng	0.9	0.1	0.346	0.454	0.279	0.548	0.315	0.283	0.196
qi-mse-ng	0.9	0.2	0.345	0.454	0.279	0.547	0.314	0.283	0.195
qi-mse-ng	0.9	0.3	0.345	0.454	0.279	0.546	0.314	0.283	0.195
qi-mse-ng	0.9	0.4	0.345	0.455	0.279	0.547	0.314	0.283	0.195
qi-mse-ng	0.9	0.5	0.346	0.455	0.279	0.547	0.315	0.283	0.197
qi-mse-ng	0.9	0.6	0.346	0.455	0.279	0.547	0.315	0.284	0.197
qi-mse-ng	0.9	0.7	0.346	0.456	0.279	0.546	0.315	0.284	0.197
qi-mse-ng	0.9	0.8	0.347	0.457	0.279	0.547	0.315	0.285	0.197
qi-mse-ng	0.9	0.9	0.348	0.457	0.28	0.547	0.323	0.285	0.198
qi-mls-pre	0.1	—	0.814	0.827	0.863	0.618	0.932	0.777	0.866
qi-mls-pre	0.1	0.1	0.826	0.852	0.87	0.643	0.933	0.784	0.875
qi-mls-pre	0.1	0.2	0.826	0.852	0.871	0.643	0.931	0.785	0.875
qi-mls-pre	0.1	0.3	0.825	0.852	0.871	0.643	0.931	0.775	0.875
qi-mls-pre	0.1	0.4	0.826	0.852	0.872	0.639	0.931	0.788	0.875
qi-mls-pre	0.1	0.5	0.809	0.852	0.872	0.639	0.93	0.684	0.876
qi-mls-pre	0.1	0.6	0.81	0.855	0.872	0.639	0.93	0.686	0.876
qi-mls-pre	0.1	0.7	0.81	0.855	0.872	0.639	0.93	0.686	0.876
qi-mls-pre	0.1	0.8	0.809	0.855	0.868	0.639	0.93	0.686	0.876
qi-mls-pre	0.1	0.9	0.809	0.855	0.869	0.638	0.93	0.687	0.876
qi-mls-pre	0.3	—	0.941	0.949	0.939	0.959	0.934	0.912	0.955
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 17 (RAND on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre	0.3	0.1	0.951	0.953	0.939	0.979	0.939	0.938	0.958
qi-mls-pre	0.3	0.2	0.951	0.953	0.938	0.979	0.938	0.937	0.957
qi-mls-pre	0.3	0.3	0.951	0.953	0.938	0.979	0.938	0.937	0.958
qi-mls-pre	0.3	0.4	0.951	0.953	0.938	0.979	0.938	0.937	0.958
qi-mls-pre	0.3	0.5	0.95	0.953	0.938	0.979	0.938	0.937	0.958
qi-mls-pre	0.3	0.6	0.95	0.953	0.937	0.979	0.938	0.937	0.958
qi-mls-pre	0.3	0.7	0.95	0.953	0.937	0.979	0.938	0.937	0.958
qi-mls-pre	0.3	0.8	0.95	0.953	0.937	0.979	0.938	0.937	0.958
qi-mls-pre	0.3	0.9	0.95	0.953	0.937	0.979	0.938	0.936	0.958
qi-mls-pre	0.5	—	0.955	0.952	0.958	0.967	0.942	0.948	0.962
qi-mls-pre	0.5	0.1	0.961	0.956	0.956	0.985	0.942	0.963	0.966
qi-mls-pre	0.5	0.2	0.961	0.955	0.956	0.985	0.942	0.962	0.965
qi-mls-pre	0.5	0.3	0.961	0.955	0.955	0.985	0.942	0.962	0.966
qi-mls-pre	0.5	0.4	0.961	0.955	0.955	0.985	0.942	0.962	0.966
qi-mls-pre	0.5	0.5	0.961	0.955	0.955	0.985	0.941	0.962	0.966
qi-mls-pre	0.5	0.6	0.961	0.955	0.955	0.984	0.941	0.962	0.966
qi-mls-pre	0.5	0.7	0.961	0.955	0.955	0.984	0.941	0.962	0.966
qi-mls-pre	0.5	0.8	0.96	0.955	0.955	0.984	0.941	0.961	0.966
qi-mls-pre	0.5	0.9	0.96	0.955	0.955	0.984	0.941	0.961	0.966
qi-mls-pre	0.7	—	0.955	0.952	0.958	0.967	0.942	0.948	0.962
qi-mls-pre	0.7	0.1	0.961	0.956	0.956	0.985	0.942	0.963	0.966
qi-mls-pre	0.7	0.2	0.961	0.955	0.956	0.985	0.942	0.962	0.965
qi-mls-pre	0.7	0.3	0.961	0.955	0.955	0.985	0.942	0.962	0.966
qi-mls-pre	0.7	0.4	0.961	0.955	0.955	0.985	0.942	0.962	0.966
qi-mls-pre	0.7	0.5	0.961	0.955	0.955	0.985	0.941	0.962	0.966
qi-mls-pre	0.7	0.6	0.961	0.955	0.955	0.984	0.941	0.962	0.966
qi-mls-pre	0.7	0.7	0.961	0.955	0.955	0.984	0.941	0.962	0.966
qi-mls-pre	0.7	0.8	0.96	0.955	0.955	0.984	0.941	0.961	0.966
qi-mls-pre	0.7	0.9	0.96	0.955	0.955	0.984	0.941	0.961	0.966
qi-mls-pre	0.9	—	0.955	0.952	0.958	0.967	0.942	0.948	0.962
qi-mls-pre	0.9	0.1	0.961	0.956	0.956	0.985	0.942	0.963	0.966
qi-mls-pre	0.9	0.2	0.961	0.955	0.956	0.985	0.942	0.962	0.965
qi-mls-pre	0.9	0.3	0.961	0.955	0.955	0.985	0.942	0.962	0.966
qi-mls-pre	0.9	0.4	0.961	0.955	0.955	0.985	0.942	0.962	0.966
qi-mls-pre	0.9	0.5	0.961	0.955	0.955	0.985	0.941	0.962	0.966
qi-mls-pre	0.9	0.6	0.961	0.955	0.955	0.984	0.941	0.962	0.966
qi-mls-pre	0.9	0.7	0.961	0.955	0.955	0.984	0.941	0.962	0.966
qi-mls-pre	0.9	0.8	0.96	0.955	0.955	0.984	0.941	0.961	0.966
qi-mls-pre	0.9	0.9	0.96	0.955	0.955	0.984	0.941	0.961	0.966
qi-mls-pre-ng	0.1	—	0.898	0.952	0.916	0.771	0.951	0.884	0.914
qi-mls-pre-ng	0.1	0.1	0.893	0.957	0.919	0.737	0.945	0.882	0.915
qi-mls-pre-ng	0.1	0.2	0.891	0.959	0.921	0.721	0.944	0.886	0.914
qi-mls-pre-ng	0.1	0.3	0.889	0.958	0.917	0.715	0.944	0.887	0.914
qi-mls-pre-ng	0.1	0.4	0.889	0.958	0.918	0.716	0.944	0.885	0.914
qi-mls-pre-ng	0.1	0.5	0.889	0.958	0.918	0.716	0.944	0.885	0.914
qi-mls-pre-ng	0.1	0.6	0.889	0.958	0.918	0.716	0.944	0.885	0.914
qi-mls-pre-ng	0.1	0.7	0.889	0.958	0.918	0.716	0.944	0.886	0.914
qi-mls-pre-ng	0.1	0.8	0.889	0.958	0.918	0.719	0.944	0.884	0.914
qi-mls-pre-ng	0.1	0.9	0.89	0.958	0.919	0.722	0.943	0.89	0.911
qi-mls-pre-ng	0.3	—	0.764	0.789	0.712	0.786	0.764	0.745	0.79
qi-mls-pre-ng	0.3	0.1	0.772	0.778	0.713	0.846	0.744	0.783	0.766
qi-mls-pre-ng	0.3	0.2	0.771	0.779	0.713	0.845	0.744	0.778	0.766
qi-mls-pre-ng	0.3	0.3	0.771	0.779	0.713	0.846	0.744	0.777	0.766
qi-mls-pre-ng	0.3	0.4	0.771	0.781	0.713	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.3	0.5	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.3	0.6	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.3	0.7	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.3	0.8	0.772	0.781	0.714	0.847	0.744	0.778	0.766
qi-mls-pre-ng	0.3	0.9	0.768	0.781	0.714	0.847	0.744	0.756	0.766
qi-mls-pre-ng	0.5	—	0.764	0.789	0.712	0.786	0.764	0.745	0.79
qi-mls-pre-ng	0.5	0.1	0.772	0.778	0.713	0.846	0.744	0.783	0.766
qi-mls-pre-ng	0.5	0.2	0.771	0.779	0.713	0.845	0.744	0.778	0.766
qi-mls-pre-ng	0.5	0.3	0.771	0.779	0.713	0.846	0.744	0.777	0.766
qi-mls-pre-ng	0.5	0.4	0.771	0.781	0.713	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.5	0.5	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.5	0.6	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.5	0.7	0.771	0.781	0.714	0.846	0.744	0.778	0.766
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 17 (RAND on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre-ng	0.5	0.8	0.772	0.781	0.714	0.847	0.744	0.778	0.766
qi-mls-pre-ng	0.5	0.9	0.768	0.781	0.714	0.847	0.744	0.756	0.766
qi-mls-pre-ng	0.7	—	0.764	0.789	0.712	0.786	0.764	0.745	0.79
qi-mls-pre-ng	0.7	0.1	0.772	0.778	0.713	0.846	0.744	0.783	0.766
qi-mls-pre-ng	0.7	0.2	0.771	0.779	0.713	0.845	0.744	0.778	0.766
qi-mls-pre-ng	0.7	0.3	0.771	0.779	0.713	0.846	0.744	0.777	0.766
qi-mls-pre-ng	0.7	0.4	0.771	0.781	0.713	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.7	0.5	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.7	0.6	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.7	0.7	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.7	0.8	0.772	0.781	0.714	0.847	0.744	0.778	0.766
qi-mls-pre-ng	0.7	0.9	0.768	0.781	0.714	0.847	0.744	0.756	0.766
qi-mls-pre-ng	0.9	—	0.764	0.789	0.712	0.786	0.764	0.745	0.79
qi-mls-pre-ng	0.9	0.1	0.772	0.778	0.713	0.846	0.744	0.783	0.766
qi-mls-pre-ng	0.9	0.2	0.771	0.779	0.713	0.845	0.744	0.778	0.766
qi-mls-pre-ng	0.9	0.3	0.771	0.779	0.713	0.846	0.744	0.777	0.766
qi-mls-pre-ng	0.9	0.4	0.771	0.781	0.713	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.9	0.5	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.9	0.6	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.9	0.7	0.771	0.781	0.714	0.846	0.744	0.778	0.766
qi-mls-pre-ng	0.9	0.8	0.772	0.781	0.714	0.847	0.744	0.778	0.766
qi-mls-pre-ng	0.9	0.9	0.768	0.781	0.714	0.847	0.744	0.756	0.766
qi-mls	0.1	—	0.247	0.158	0.146	0.488	0.216	0.325	0.151
qi-mls	0.1	0.1	0.23	0.159	0.144	0.525	0.179	0.242	0.131
qi-mls	0.1	0.2	0.229	0.161	0.144	0.525	0.178	0.239	0.129
qi-mls	0.1	0.3	0.229	0.161	0.142	0.525	0.178	0.237	0.128
qi-mls	0.1	0.4	0.228	0.163	0.142	0.525	0.178	0.236	0.127
qi-mls	0.1	0.5	0.228	0.163	0.141	0.525	0.179	0.236	0.126
qi-mls	0.1	0.6	0.228	0.163	0.142	0.525	0.177	0.235	0.126
qi-mls	0.1	0.7	0.228	0.163	0.142	0.525	0.177	0.235	0.126
qi-mls	0.1	0.8	0.229	0.163	0.142	0.525	0.178	0.243	0.126
qi-mls	0.1	0.9	0.23	0.164	0.142	0.525	0.178	0.245	0.126
qi-mls	0.3	—	0.505	0.535	0.479	0.597	0.537	0.529	0.352
qi-mls	0.3	0.1	0.497	0.567	0.449	0.649	0.528	0.436	0.356
qi-mls	0.3	0.2	0.498	0.567	0.449	0.649	0.529	0.44	0.356
qi-mls	0.3	0.3	0.498	0.567	0.448	0.649	0.531	0.44	0.355
qi-mls	0.3	0.4	0.499	0.568	0.448	0.649	0.531	0.44	0.356
qi-mls	0.3	0.5	0.499	0.568	0.448	0.651	0.532	0.44	0.356
qi-mls	0.3	0.6	0.5	0.569	0.448	0.651	0.532	0.441	0.356
qi-mls	0.3	0.7	0.5	0.569	0.448	0.651	0.532	0.442	0.356
qi-mls	0.3	0.8	0.5	0.569	0.449	0.651	0.533	0.443	0.357
qi-mls	0.3	0.9	0.502	0.57	0.45	0.652	0.535	0.445	0.358
qi-mls	0.5	—	0.968	0.981	0.998	0.924	0.97	0.943	0.991
qi-mls	0.5	0.1	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.5	0.2	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.5	0.3	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.5	0.4	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.5	0.5	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.5	0.6	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.5	0.7	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.5	0.8	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.5	0.9	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	—	0.968	0.981	0.998	0.924	0.97	0.943	0.991
qi-mls	0.7	0.1	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	0.2	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	0.3	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	0.4	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	0.5	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	0.6	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	0.7	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	0.8	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.7	0.9	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.9	—	0.968	0.981	0.998	0.924	0.97	0.943	0.991
qi-mls	0.9	0.1	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.9	0.2	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.9	0.3	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.9	0.4	0.999	0.998	0.999	1.0	0.999	1.0	0.999
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 17 (RAND on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls	0.9	0.5	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.9	0.6	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.9	0.7	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.9	0.8	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls	0.9	0.9	0.999	0.998	0.999	1.0	0.999	1.0	0.999
qi-mls-ng	0.1	—	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.1	0.1	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.1	0.2	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.1	0.3	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.1	0.4	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.1	0.5	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.1	0.6	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.1	0.7	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.1	0.8	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.1	0.9	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.3	—	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.3	0.1	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.3	0.2	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.3	0.3	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.3	0.4	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.3	0.5	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.3	0.6	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.3	0.7	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.3	0.8	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.3	0.9	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.5	—	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.5	0.1	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.5	0.2	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.5	0.3	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.5	0.4	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.5	0.5	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.5	0.6	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.5	0.7	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.5	0.8	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.5	0.9	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.7	—	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.7	0.1	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.7	0.2	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.7	0.3	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.7	0.4	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.7	0.5	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.7	0.6	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.7	0.7	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.7	0.8	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.7	0.9	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.9	—	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.9	0.1	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.9	0.2	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.9	0.3	0.991	1.0	1.0	1.0	0.982	0.999	0.963
qi-mls-ng	0.9	0.4	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.9	0.5	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.9	0.6	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.9	0.7	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.9	0.8	0.99	1.0	1.0	1.0	0.979	0.999	0.963
qi-mls-ng	0.9	0.9	0.99	1.0	1.0	1.0	0.979	0.999	0.963
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Table 18: CREMI score performance on 25% of the data that were not used for training for all architectures and parameter sets. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging.

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	—	2.283	2.487	2.333	2.016	2.413	2.08	2.366
qi-mse	0.1	0.1	2.212	2.462	2.324	1.9	2.403	1.855	2.329
qi-mse	0.1	0.2	2.209	2.461	2.321	1.897	2.401	1.849	2.326
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 18 (CREMI score on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	0.3	2.221	2.461	2.319	1.897	2.4	1.923	2.325
qi-mse	0.1	0.4	2.22	2.461	2.318	1.897	2.4	1.923	2.324
qi-mse	0.1	0.5	2.221	2.461	2.317	1.898	2.4	1.923	2.324
qi-mse	0.1	0.6	2.221	2.461	2.317	1.9	2.4	1.924	2.324
qi-mse	0.1	0.7	2.222	2.462	2.317	1.902	2.4	1.926	2.324
qi-mse	0.1	0.8	2.222	2.462	2.317	1.905	2.4	1.928	2.324
qi-mse	0.1	0.9	2.225	2.463	2.317	1.911	2.4	1.932	2.324
qi-mse	0.3	—	2.296	2.489	2.328	2.108	2.414	2.074	2.362
qi-mse	0.3	0.1	2.203	2.462	2.317	1.868	2.393	1.854	2.326
qi-mse	0.3	0.2	2.205	2.461	2.313	1.895	2.391	1.848	2.323
qi-mse	0.3	0.3	2.199	2.461	2.311	1.865	2.39	1.846	2.321
qi-mse	0.3	0.4	2.199	2.461	2.31	1.865	2.389	1.846	2.32
qi-mse	0.3	0.5	2.199	2.461	2.31	1.866	2.389	1.847	2.32
qi-mse	0.3	0.6	2.199	2.461	2.309	1.868	2.389	1.848	2.32
qi-mse	0.3	0.7	2.2	2.461	2.309	1.87	2.389	1.849	2.32
qi-mse	0.3	0.8	2.201	2.462	2.308	1.872	2.389	1.852	2.32
qi-mse	0.3	0.9	2.203	2.463	2.309	1.879	2.389	1.856	2.32
qi-mse	0.5	—	1.078	0.885	1.441	1.201	0.752	1.436	0.754
qi-mse	0.5	0.1	0.935	0.805	1.631	0.871	0.799	0.776	0.726
qi-mse	0.5	0.2	0.919	0.778	1.615	0.863	0.793	0.762	0.705
qi-mse	0.5	0.3	0.927	0.795	1.607	0.861	0.766	0.883	0.649
qi-mse	0.5	0.4	0.936	0.778	1.6	0.858	0.778	0.881	0.723
qi-mse	0.5	0.5	0.907	0.843	1.599	0.859	0.731	0.756	0.653
qi-mse	0.5	0.6	0.875	0.792	1.427	0.866	0.724	0.783	0.655
qi-mse	0.5	0.7	0.871	0.752	1.427	0.868	0.723	0.785	0.67
qi-mse	0.5	0.8	0.882	0.733	1.428	0.872	0.723	0.866	0.671
qi-mse	0.5	0.9	0.856	0.78	1.428	0.881	0.606	0.77	0.671
qi-mse	0.7	—	0.565	0.186	0.626	1.206	0.323	0.682	0.368
qi-mse	0.7	0.1	0.438	0.137	0.61	0.855	0.378	0.336	0.31
qi-mse	0.7	0.2	0.426	0.134	0.603	0.85	0.339	0.324	0.305
qi-mse	0.7	0.3	0.425	0.136	0.6	0.848	0.338	0.321	0.306
qi-mse	0.7	0.4	0.425	0.136	0.598	0.855	0.337	0.321	0.305
qi-mse	0.7	0.5	0.426	0.137	0.597	0.856	0.336	0.322	0.305
qi-mse	0.7	0.6	0.426	0.137	0.597	0.858	0.335	0.325	0.305
qi-mse	0.7	0.7	0.428	0.137	0.599	0.861	0.337	0.328	0.305
qi-mse	0.7	0.8	0.43	0.137	0.599	0.867	0.341	0.333	0.306
qi-mse	0.7	0.9	0.435	0.138	0.601	0.877	0.341	0.344	0.307
qi-mse	0.9	—	1.116	1.459	0.859	1.469	1.107	0.961	0.841
qi-mse	0.9	0.1	0.998	1.407	0.85	1.148	1.08	0.668	0.835
qi-mse	0.9	0.2	0.994	1.406	0.845	1.144	1.076	0.662	0.833
qi-mse	0.9	0.3	0.995	1.406	0.844	1.143	1.083	0.661	0.833
qi-mse	0.9	0.4	0.995	1.407	0.844	1.144	1.083	0.659	0.833
qi-mse	0.9	0.5	0.995	1.407	0.843	1.146	1.082	0.66	0.833
qi-mse	0.9	0.6	0.999	1.408	0.843	1.147	1.082	0.679	0.836
qi-mse	0.9	0.7	1.001	1.409	0.844	1.15	1.082	0.683	0.837
qi-mse	0.9	0.8	1.003	1.411	0.845	1.154	1.084	0.688	0.838
qi-mse	0.9	0.9	1.008	1.412	0.846	1.165	1.091	0.696	0.84
qi-mse-ng	0.1	—	2.278	2.487	2.333	2.014	2.412	2.056	2.365
qi-mse-ng	0.1	0.1	2.218	2.461	2.32	1.896	2.37	1.933	2.328
qi-mse-ng	0.1	0.2	2.216	2.46	2.318	1.894	2.369	1.93	2.326
qi-mse-ng	0.1	0.3	2.215	2.46	2.317	1.894	2.368	1.929	2.324
qi-mse-ng	0.1	0.4	2.215	2.46	2.316	1.895	2.367	1.929	2.324
qi-mse-ng	0.1	0.5	2.218	2.46	2.316	1.915	2.367	1.93	2.323
qi-mse-ng	0.1	0.6	2.219	2.46	2.315	1.916	2.367	1.931	2.323
qi-mse-ng	0.1	0.7	2.219	2.46	2.315	1.919	2.367	1.933	2.322
qi-mse-ng	0.1	0.8	2.221	2.46	2.315	1.922	2.367	1.936	2.322
qi-mse-ng	0.1	0.9	2.22	2.46	2.316	1.911	2.368	1.943	2.323
qi-mse-ng	0.3	—	2.304	2.487	2.333	2.11	2.415	2.112	2.365
qi-mse-ng	0.3	0.1	2.218	2.461	2.32	1.895	2.37	1.933	2.328
qi-mse-ng	0.3	0.2	2.216	2.46	2.316	1.894	2.369	1.93	2.325
qi-mse-ng	0.3	0.3	2.215	2.46	2.315	1.894	2.368	1.929	2.324
qi-mse-ng	0.3	0.4	2.214	2.46	2.314	1.894	2.367	1.929	2.323
qi-mse-ng	0.3	0.5	2.214	2.46	2.313	1.895	2.366	1.93	2.322
qi-mse-ng	0.3	0.6	2.215	2.46	2.313	1.896	2.366	1.931	2.322
qi-mse-ng	0.3	0.7	2.215	2.46	2.313	1.898	2.366	1.933	2.321
qi-mse-ng	0.3	0.8	2.194	2.46	2.313	1.768	2.367	1.937	2.321
qi-mse-ng	0.3	0.9	2.197	2.46	2.313	1.776	2.368	1.944	2.322
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 18 (CREMI score on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse-ng	0.5	—	1.903	1.694	2.089	1.455	2.186	1.984	2.011
qi-mse-ng	0.5	0.1	1.816	1.666	2.096	1.209	2.126	1.738	2.065
qi-mse-ng	0.5	0.2	1.802	1.603	2.092	1.195	2.142	1.731	2.051
qi-mse-ng	0.5	0.3	1.812	1.63	2.091	1.227	2.141	1.729	2.056
qi-mse-ng	0.5	0.4	1.807	1.635	2.091	1.204	2.14	1.728	2.046
qi-mse-ng	0.5	0.5	1.809	1.669	2.09	1.191	2.116	1.742	2.045
qi-mse-ng	0.5	0.6	1.804	1.654	2.09	1.192	2.116	1.731	2.044
qi-mse-ng	0.5	0.7	1.785	1.599	2.089	1.148	2.099	1.732	2.044
qi-mse-ng	0.5	0.8	1.794	1.62	2.089	1.181	2.099	1.735	2.042
qi-mse-ng	0.5	0.9	1.799	1.628	2.087	1.192	2.1	1.744	2.043
qi-mse-ng	0.7	—	0.587	0.222	0.638	1.181	0.379	0.682	0.423
qi-mse-ng	0.7	0.1	0.439	0.175	0.616	0.864	0.283	0.364	0.335
qi-mse-ng	0.7	0.2	0.438	0.173	0.613	0.855	0.302	0.356	0.329
qi-mse-ng	0.7	0.3	0.433	0.173	0.611	0.856	0.278	0.354	0.328
qi-mse-ng	0.7	0.4	0.436	0.173	0.607	0.856	0.304	0.353	0.326
qi-mse-ng	0.7	0.5	0.437	0.173	0.607	0.859	0.304	0.355	0.325
qi-mse-ng	0.7	0.6	0.438	0.173	0.606	0.861	0.303	0.358	0.325
qi-mse-ng	0.7	0.7	0.44	0.173	0.606	0.865	0.304	0.365	0.326
qi-mse-ng	0.7	0.8	0.443	0.173	0.608	0.87	0.306	0.375	0.327
qi-mse-ng	0.7	0.9	0.45	0.177	0.61	0.885	0.308	0.39	0.328
qi-mse-ng	0.9	—	0.995	1.244	0.769	1.404	0.921	0.887	0.747
qi-mse-ng	0.9	0.1	0.859	1.195	0.755	1.102	0.826	0.597	0.683
qi-mse-ng	0.9	0.2	0.857	1.196	0.752	1.097	0.825	0.594	0.68
qi-mse-ng	0.9	0.3	0.857	1.196	0.751	1.097	0.824	0.594	0.678
qi-mse-ng	0.9	0.4	0.858	1.198	0.751	1.1	0.824	0.596	0.678
qi-mse-ng	0.9	0.5	0.859	1.198	0.75	1.101	0.824	0.598	0.681
qi-mse-ng	0.9	0.6	0.86	1.2	0.751	1.103	0.825	0.602	0.682
qi-mse-ng	0.9	0.7	0.862	1.203	0.752	1.106	0.825	0.606	0.682
qi-mse-ng	0.9	0.8	0.865	1.205	0.753	1.111	0.826	0.612	0.683
qi-mse-ng	0.9	0.9	0.872	1.207	0.754	1.123	0.841	0.624	0.686
qi-mls-pre	0.1	—	1.808	1.718	1.918	1.524	2.09	1.634	1.966
qi-mls-pre	0.1	0.1	1.779	1.767	1.957	1.38	2.068	1.537	1.966
qi-mls-pre	0.1	0.2	1.773	1.765	1.949	1.373	2.057	1.533	1.963
qi-mls-pre	0.1	0.3	1.767	1.765	1.952	1.371	2.053	1.498	1.966
qi-mls-pre	0.1	0.4	1.771	1.765	1.95	1.359	2.052	1.535	1.967
qi-mls-pre	0.1	0.5	1.742	1.765	1.948	1.36	2.049	1.363	1.965
qi-mls-pre	0.1	0.6	1.744	1.775	1.947	1.362	2.048	1.366	1.965
qi-mls-pre	0.1	0.7	1.744	1.772	1.946	1.365	2.049	1.37	1.964
qi-mls-pre	0.1	0.8	1.744	1.772	1.925	1.373	2.05	1.38	1.964
qi-mls-pre	0.1	0.9	1.75	1.773	1.925	1.385	2.052	1.402	1.964
qi-mls-pre	0.3	—	2.541	2.601	2.49	2.612	2.498	2.435	2.609
qi-mls-pre	0.3	0.1	2.458	2.568	2.467	2.416	2.442	2.284	2.57
qi-mls-pre	0.3	0.2	2.455	2.567	2.462	2.414	2.44	2.281	2.566
qi-mls-pre	0.3	0.3	2.455	2.567	2.459	2.414	2.439	2.281	2.569
qi-mls-pre	0.3	0.4	2.455	2.567	2.457	2.416	2.438	2.283	2.567
qi-mls-pre	0.3	0.5	2.455	2.567	2.454	2.418	2.438	2.287	2.567
qi-mls-pre	0.3	0.6	2.456	2.567	2.453	2.421	2.439	2.291	2.566
qi-mls-pre	0.3	0.7	2.458	2.567	2.451	2.426	2.439	2.297	2.566
qi-mls-pre	0.3	0.8	2.461	2.567	2.451	2.433	2.44	2.307	2.565
qi-mls-pre	0.3	0.9	2.466	2.567	2.45	2.446	2.444	2.324	2.565
qi-mls-pre	0.5	—	2.644	2.645	2.628	2.7	2.555	2.662	2.676
qi-mls-pre	0.5	0.1	2.542	2.603	2.589	2.494	2.478	2.456	2.631
qi-mls-pre	0.5	0.2	2.539	2.602	2.584	2.492	2.476	2.453	2.628
qi-mls-pre	0.5	0.3	2.539	2.602	2.581	2.492	2.474	2.453	2.63
qi-mls-pre	0.5	0.4	2.539	2.602	2.579	2.494	2.474	2.455	2.629
qi-mls-pre	0.5	0.5	2.539	2.602	2.577	2.496	2.474	2.458	2.628
qi-mls-pre	0.5	0.6	2.54	2.602	2.575	2.499	2.474	2.462	2.627
qi-mls-pre	0.5	0.7	2.542	2.602	2.574	2.504	2.474	2.468	2.627
qi-mls-pre	0.5	0.8	2.544	2.602	2.573	2.51	2.476	2.477	2.627
qi-mls-pre	0.5	0.9	2.549	2.602	2.572	2.523	2.479	2.493	2.626
qi-mls-pre	0.7	—	2.644	2.645	2.628	2.7	2.555	2.662	2.676
qi-mls-pre	0.7	0.1	2.542	2.603	2.589	2.494	2.478	2.456	2.631
qi-mls-pre	0.7	0.2	2.539	2.602	2.584	2.492	2.476	2.453	2.628
qi-mls-pre	0.7	0.3	2.539	2.602	2.581	2.492	2.474	2.453	2.63
qi-mls-pre	0.7	0.4	2.539	2.602	2.579	2.494	2.474	2.455	2.629
qi-mls-pre	0.7	0.5	2.539	2.602	2.577	2.496	2.474	2.458	2.628
qi-mls-pre	0.7	0.6	2.54	2.602	2.575	2.499	2.474	2.462	2.627
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 18 (CREMI score on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre	0.7	0.7	2.542	2.602	2.574	2.504	2.474	2.468	2.627
qi-mls-pre	0.7	0.8	2.544	2.602	2.573	2.51	2.476	2.477	2.627
qi-mls-pre	0.7	0.9	2.549	2.602	2.572	2.523	2.479	2.493	2.626
qi-mls-pre	0.9	—	2.644	2.645	2.628	2.7	2.555	2.662	2.676
qi-mls-pre	0.9	0.1	2.542	2.603	2.589	2.494	2.478	2.456	2.631
qi-mls-pre	0.9	0.2	2.539	2.602	2.584	2.492	2.476	2.453	2.628
qi-mls-pre	0.9	0.3	2.539	2.602	2.581	2.492	2.474	2.453	2.63
qi-mls-pre	0.9	0.4	2.539	2.602	2.579	2.494	2.474	2.455	2.629
qi-mls-pre	0.9	0.5	2.539	2.602	2.577	2.496	2.474	2.458	2.628
qi-mls-pre	0.9	0.6	2.54	2.602	2.575	2.499	2.474	2.462	2.627
qi-mls-pre	0.9	0.7	2.542	2.602	2.574	2.504	2.474	2.468	2.627
qi-mls-pre	0.9	0.8	2.544	2.602	2.573	2.51	2.476	2.477	2.627
qi-mls-pre	0.9	0.9	2.549	2.602	2.572	2.523	2.479	2.493	2.626
qi-mls-pre-ng	0.1	—	2.187	2.303	2.222	1.923	2.315	2.052	2.309
qi-mls-pre-ng	0.1	0.1	2.133	2.329	2.25	1.703	2.273	1.926	2.316
qi-mls-pre-ng	0.1	0.2	2.127	2.332	2.253	1.677	2.262	1.932	2.307
qi-mls-pre-ng	0.1	0.3	2.12	2.327	2.226	1.668	2.261	1.932	2.303
qi-mls-pre-ng	0.1	0.4	2.117	2.325	2.224	1.669	2.26	1.92	2.303
qi-mls-pre-ng	0.1	0.5	2.117	2.325	2.223	1.67	2.26	1.922	2.302
qi-mls-pre-ng	0.1	0.6	2.117	2.325	2.223	1.672	2.259	1.923	2.302
qi-mls-pre-ng	0.1	0.7	2.118	2.325	2.224	1.674	2.259	1.924	2.302
qi-mls-pre-ng	0.1	0.8	2.121	2.326	2.227	1.69	2.259	1.922	2.302
qi-mls-pre-ng	0.1	0.9	2.128	2.328	2.228	1.708	2.252	1.952	2.3
qi-mls-pre-ng	0.3	—	1.871	1.868	1.774	1.922	1.872	1.791	1.999
qi-mls-pre-ng	0.3	0.1	1.811	1.829	1.767	1.832	1.795	1.705	1.937
qi-mls-pre-ng	0.3	0.2	1.803	1.83	1.76	1.826	1.79	1.679	1.933
qi-mls-pre-ng	0.3	0.3	1.803	1.829	1.757	1.833	1.789	1.678	1.933
qi-mls-pre-ng	0.3	0.4	1.803	1.834	1.755	1.834	1.789	1.673	1.933
qi-mls-pre-ng	0.3	0.5	1.804	1.834	1.757	1.834	1.789	1.675	1.933
qi-mls-pre-ng	0.3	0.6	1.804	1.834	1.757	1.837	1.788	1.674	1.932
qi-mls-pre-ng	0.3	0.7	1.805	1.835	1.756	1.839	1.788	1.677	1.933
qi-mls-pre-ng	0.3	0.8	1.807	1.835	1.756	1.847	1.788	1.68	1.933
qi-mls-pre-ng	0.3	0.9	1.803	1.837	1.758	1.858	1.789	1.644	1.933
qi-mls-pre-ng	0.5	—	1.871	1.868	1.774	1.922	1.872	1.791	2.001
qi-mls-pre-ng	0.5	0.1	1.811	1.829	1.767	1.832	1.795	1.705	1.938
qi-mls-pre-ng	0.5	0.2	1.803	1.83	1.76	1.826	1.79	1.679	1.934
qi-mls-pre-ng	0.5	0.3	1.803	1.829	1.757	1.833	1.789	1.678	1.934
qi-mls-pre-ng	0.5	0.4	1.803	1.834	1.755	1.834	1.789	1.673	1.934
qi-mls-pre-ng	0.5	0.5	1.804	1.834	1.757	1.834	1.789	1.675	1.933
qi-mls-pre-ng	0.5	0.6	1.804	1.834	1.757	1.837	1.788	1.674	1.933
qi-mls-pre-ng	0.5	0.7	1.805	1.835	1.756	1.839	1.788	1.677	1.933
qi-mls-pre-ng	0.5	0.8	1.807	1.835	1.756	1.847	1.788	1.68	1.933
qi-mls-pre-ng	0.5	0.9	1.803	1.837	1.758	1.858	1.789	1.644	1.934
qi-mls-pre-ng	0.7	—	1.871	1.868	1.774	1.922	1.872	1.791	2.001
qi-mls-pre-ng	0.7	0.1	1.811	1.829	1.767	1.832	1.795	1.705	1.938
qi-mls-pre-ng	0.7	0.2	1.803	1.83	1.76	1.826	1.79	1.679	1.934
qi-mls-pre-ng	0.7	0.3	1.803	1.829	1.757	1.833	1.789	1.678	1.934
qi-mls-pre-ng	0.7	0.4	1.803	1.834	1.755	1.834	1.789	1.673	1.934
qi-mls-pre-ng	0.7	0.5	1.804	1.834	1.757	1.834	1.789	1.675	1.933
qi-mls-pre-ng	0.7	0.6	1.804	1.834	1.757	1.837	1.788	1.674	1.933
qi-mls-pre-ng	0.7	0.7	1.805	1.835	1.756	1.839	1.788	1.677	1.933
qi-mls-pre-ng	0.7	0.8	1.807	1.835	1.756	1.847	1.788	1.68	1.933
qi-mls-pre-ng	0.7	0.9	1.803	1.837	1.758	1.858	1.789	1.644	1.934
qi-mls-pre-ng	0.9	—	1.871	1.868	1.774	1.922	1.872	1.791	2.001
qi-mls-pre-ng	0.9	0.1	1.811	1.829	1.767	1.832	1.795	1.705	1.938
qi-mls-pre-ng	0.9	0.2	1.803	1.83	1.76	1.826	1.79	1.679	1.934
qi-mls-pre-ng	0.9	0.3	1.803	1.829	1.757	1.833	1.789	1.678	1.934
qi-mls-pre-ng	0.9	0.4	1.803	1.834	1.755	1.834	1.789	1.673	1.934
qi-mls-pre-ng	0.9	0.5	1.804	1.834	1.757	1.834	1.789	1.675	1.933
qi-mls-pre-ng	0.9	0.6	1.804	1.834	1.757	1.837	1.788	1.674	1.933
qi-mls-pre-ng	0.9	0.7	1.805	1.835	1.756	1.839	1.788	1.677	1.933
qi-mls-pre-ng	0.9	0.8	1.807	1.835	1.756	1.847	1.788	1.68	1.933
qi-mls-pre-ng	0.9	0.9	1.803	1.837	1.758	1.858	1.789	1.644	1.934
qi-mls	0.1	—	0.611	0.397	0.479	1.077	0.545	0.673	0.496
qi-mls	0.1	0.1	0.518	0.382	0.468	0.95	0.459	0.404	0.443
qi-mls	0.1	0.2	0.514	0.387	0.464	0.95	0.456	0.389	0.437
qi-mls	0.1	0.3	0.511	0.387	0.459	0.948	0.453	0.382	0.434
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 18 (CREMI score on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls	0.1	0.4	0.509	0.39	0.458	0.948	0.452	0.378	0.429
qi-mls	0.1	0.5	0.509	0.389	0.457	0.949	0.455	0.375	0.426
qi-mls	0.1	0.6	0.508	0.39	0.457	0.952	0.451	0.374	0.428
qi-mls	0.1	0.7	0.509	0.392	0.458	0.953	0.45	0.376	0.427
qi-mls	0.1	0.8	0.513	0.392	0.457	0.957	0.452	0.391	0.427
qi-mls	0.1	0.9	0.518	0.393	0.459	0.966	0.454	0.408	0.428
qi-mls	0.3	—	1.328	1.292	1.327	1.521	1.383	1.328	1.115
qi-mls	0.3	0.1	1.255	1.319	1.267	1.449	1.341	1.041	1.113
qi-mls	0.3	0.2	1.256	1.319	1.267	1.449	1.34	1.05	1.111
qi-mls	0.3	0.3	1.257	1.32	1.264	1.452	1.345	1.05	1.11
qi-mls	0.3	0.4	1.258	1.322	1.264	1.454	1.345	1.052	1.112
qi-mls	0.3	0.5	1.26	1.324	1.264	1.462	1.346	1.053	1.111
qi-mls	0.3	0.6	1.262	1.324	1.264	1.465	1.348	1.056	1.113
qi-mls	0.3	0.7	1.264	1.326	1.265	1.469	1.349	1.061	1.115
qi-mls	0.3	0.8	1.267	1.327	1.267	1.474	1.353	1.067	1.117
qi-mls	0.3	0.9	1.275	1.331	1.27	1.486	1.357	1.085	1.121
qi-mls	0.5	—	3.204	3.165	3.301	3.079	3.19	3.254	3.237
qi-mls	0.5	0.1	3.223	3.19	3.299	3.113	3.225	3.266	3.247
qi-mls	0.5	0.2	3.222	3.189	3.298	3.112	3.223	3.264	3.245
qi-mls	0.5	0.3	3.221	3.189	3.297	3.112	3.222	3.263	3.244
qi-mls	0.5	0.4	3.221	3.189	3.296	3.112	3.222	3.263	3.243
qi-mls	0.5	0.5	3.221	3.189	3.295	3.113	3.222	3.263	3.243
qi-mls	0.5	0.6	3.221	3.189	3.295	3.114	3.221	3.263	3.243
qi-mls	0.5	0.7	3.221	3.189	3.295	3.115	3.221	3.263	3.242
qi-mls	0.5	0.8	3.222	3.189	3.295	3.117	3.221	3.265	3.242
qi-mls	0.5	0.9	3.223	3.189	3.295	3.122	3.222	3.269	3.242
qi-mls	0.7	—	3.204	3.165	3.301	3.079	3.19	3.254	3.237
qi-mls	0.7	0.1	3.223	3.19	3.299	3.113	3.225	3.266	3.247
qi-mls	0.7	0.2	3.222	3.189	3.298	3.112	3.223	3.264	3.245
qi-mls	0.7	0.3	3.221	3.189	3.297	3.112	3.222	3.263	3.244
qi-mls	0.7	0.4	3.221	3.189	3.296	3.112	3.222	3.263	3.243
qi-mls	0.7	0.5	3.221	3.189	3.295	3.113	3.222	3.263	3.243
qi-mls	0.7	0.6	3.221	3.189	3.295	3.114	3.221	3.263	3.243
qi-mls	0.7	0.7	3.221	3.189	3.295	3.115	3.221	3.263	3.242
qi-mls	0.7	0.8	3.222	3.189	3.295	3.117	3.221	3.265	3.242
qi-mls	0.7	0.9	3.223	3.189	3.295	3.122	3.222	3.269	3.242
qi-mls	0.9	—	3.204	3.165	3.301	3.079	3.19	3.254	3.237
qi-mls	0.9	0.1	3.223	3.19	3.299	3.113	3.225	3.266	3.247
qi-mls	0.9	0.2	3.222	3.189	3.298	3.112	3.223	3.264	3.245
qi-mls	0.9	0.3	3.221	3.189	3.297	3.112	3.222	3.263	3.244
qi-mls	0.9	0.4	3.221	3.189	3.296	3.112	3.222	3.263	3.243
qi-mls	0.9	0.5	3.221	3.189	3.295	3.113	3.222	3.263	3.243
qi-mls	0.9	0.6	3.221	3.189	3.295	3.114	3.221	3.263	3.243
qi-mls	0.9	0.7	3.221	3.189	3.295	3.115	3.221	3.263	3.242
qi-mls	0.9	0.8	3.222	3.189	3.295	3.117	3.221	3.265	3.242
qi-mls	0.9	0.9	3.223	3.189	3.295	3.122	3.222	3.269	3.242
qi-mls-ng	0.1	—	2.361	2.527	2.409	2.252	2.417	2.166	2.398
qi-mls-ng	0.1	0.1	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.1	0.2	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.1	0.3	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.1	0.4	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.1	0.5	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.1	0.6	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.1	0.7	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.1	0.8	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.1	0.9	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.3	—	2.361	2.527	2.409	2.252	2.417	2.166	2.398
qi-mls-ng	0.3	0.1	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.3	0.2	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.3	0.3	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.3	0.4	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.3	0.5	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.3	0.6	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.3	0.7	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.3	0.8	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.3	0.9	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.5	—	2.361	2.527	2.409	2.252	2.417	2.166	2.398
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 18 (CREMI score on 25%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-ng	0.5	0.1	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.5	0.2	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.5	0.3	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.5	0.4	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.5	0.5	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.5	0.6	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.5	0.7	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.5	0.8	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.5	0.9	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.7	—	2.361	2.527	2.409	2.252	2.417	2.166	2.398
qi-mls-ng	0.7	0.1	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.7	0.2	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.7	0.3	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.7	0.4	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.7	0.5	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.7	0.6	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.7	0.7	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.7	0.8	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.7	0.9	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.9	—	2.361	2.527	2.409	2.252	2.417	2.166	2.398
qi-mls-ng	0.9	0.1	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.9	0.2	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.9	0.3	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.9	0.4	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.9	0.5	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.9	0.6	2.361	2.527	2.409	2.252	2.419	2.163	2.397
qi-mls-ng	0.9	0.7	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.9	0.8	2.361	2.527	2.409	2.252	2.419	2.164	2.397
qi-mls-ng	0.9	0.9	2.361	2.527	2.409	2.252	2.419	2.164	2.397
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Table 19: VOI_s performance on 100% of the data including training data for all architectures and parameter sets. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging.

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	—	0.435	0.011	0.002	0.864	0.602	0.734	0.395
qi-mse	0.1	0.1	0.162	0.017	0.062	0.772	0.06	0.033	0.028
qi-mse	0.1	0.2	0.152	0.008	0.046	0.765	0.05	0.023	0.018
qi-mse	0.1	0.3	0.147	0.005	0.038	0.763	0.041	0.02	0.015
qi-mse	0.1	0.4	0.145	0.004	0.031	0.762	0.043	0.018	0.013
qi-mse	0.1	0.5	0.145	0.004	0.028	0.762	0.042	0.018	0.013
qi-mse	0.1	0.6	0.144	0.004	0.026	0.763	0.042	0.018	0.012
qi-mse	0.1	0.7	0.144	0.004	0.024	0.764	0.042	0.019	0.012
qi-mse	0.1	0.8	0.145	0.004	0.024	0.766	0.042	0.021	0.013
qi-mse	0.1	0.9	0.148	0.005	0.026	0.77	0.045	0.027	0.016
qi-mse	0.3	—	0.632	0.082	0.071	1.039	0.88	1.125	0.598
qi-mse	0.3	0.1	0.173	0.023	0.077	0.784	0.075	0.042	0.035
qi-mse	0.3	0.2	0.163	0.014	0.061	0.777	0.066	0.032	0.026
qi-mse	0.3	0.3	0.159	0.011	0.054	0.775	0.062	0.028	0.023
qi-mse	0.3	0.4	0.157	0.01	0.05	0.775	0.06	0.028	0.022
qi-mse	0.3	0.5	0.157	0.01	0.047	0.775	0.059	0.028	0.022
qi-mse	0.3	0.6	0.157	0.01	0.046	0.776	0.059	0.03	0.022
qi-mse	0.3	0.7	0.158	0.01	0.047	0.778	0.059	0.034	0.023
qi-mse	0.3	0.8	0.161	0.011	0.05	0.782	0.061	0.039	0.024
qi-mse	0.3	0.9	0.168	0.012	0.06	0.788	0.067	0.053	0.027
qi-mse	0.5	—	1.286	0.351	0.693	1.883	1.288	2.163	1.335
qi-mse	0.5	0.1	0.475	0.175	0.48	1.37	0.157	0.319	0.346
qi-mse	0.5	0.2	0.469	0.167	0.467	1.372	0.149	0.316	0.345
qi-mse	0.5	0.3	0.469	0.165	0.471	1.373	0.146	0.315	0.343
qi-mse	0.5	0.4	0.471	0.164	0.484	1.371	0.145	0.318	0.342
qi-mse	0.5	0.5	0.474	0.164	0.486	1.373	0.149	0.322	0.347
qi-mse	0.5	0.6	0.48	0.177	0.492	1.38	0.153	0.333	0.348
qi-mse	0.5	0.7	0.482	0.167	0.498	1.384	0.154	0.339	0.35
qi-mse	0.5	0.8	0.49	0.18	0.505	1.389	0.157	0.349	0.359
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 19 (VOI_s on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.5	0.9	0.502	0.182	0.522	1.401	0.167	0.365	0.376
qi-mse	0.7	—	1.76	0.821	1.254	2.408	1.551	2.718	1.808
qi-mse	0.7	0.1	0.796	0.568	0.886	1.838	0.299	0.573	0.611
qi-mse	0.7	0.2	0.793	0.564	0.877	1.838	0.297	0.574	0.608
qi-mse	0.7	0.3	0.797	0.564	0.878	1.841	0.298	0.591	0.611
qi-mse	0.7	0.4	0.801	0.565	0.882	1.849	0.299	0.6	0.611
qi-mse	0.7	0.5	0.803	0.565	0.884	1.849	0.301	0.606	0.614
qi-mse	0.7	0.6	0.807	0.566	0.89	1.853	0.302	0.615	0.615
qi-mse	0.7	0.7	0.813	0.567	0.901	1.858	0.312	0.625	0.618
qi-mse	0.7	0.8	0.822	0.57	0.91	1.866	0.322	0.641	0.623
qi-mse	0.7	0.9	0.837	0.578	0.929	1.879	0.332	0.669	0.634
qi-mse	0.9	—	3.322	4.133	2.587	3.787	2.519	3.789	3.119
qi-mse	0.9	0.1	2.309	3.854	2.172	3.206	1.172	1.609	1.844
qi-mse	0.9	0.2	2.316	3.861	2.176	3.209	1.174	1.62	1.858
qi-mse	0.9	0.3	2.322	3.867	2.183	3.212	1.181	1.627	1.862
qi-mse	0.9	0.4	2.326	3.876	2.189	3.214	1.183	1.63	1.866
qi-mse	0.9	0.5	2.332	3.882	2.193	3.217	1.187	1.636	1.876
qi-mse	0.9	0.6	2.339	3.888	2.198	3.224	1.192	1.653	1.881
qi-mse	0.9	0.7	2.347	3.897	2.206	3.228	1.196	1.665	1.888
qi-mse	0.9	0.8	2.357	3.903	2.219	3.234	1.204	1.684	1.901
qi-mse	0.9	0.9	2.371	3.911	2.238	3.246	1.216	1.705	1.909
qi-mse-ng	0.1	—	0.299	0.014	0.001	0.859	0.274	0.315	0.33
qi-mse-ng	0.1	0.1	0.152	0.012	0.046	0.779	0.019	0.032	0.023
qi-mse-ng	0.1	0.2	0.145	0.006	0.035	0.774	0.012	0.026	0.016
qi-mse-ng	0.1	0.3	0.142	0.004	0.031	0.772	0.009	0.023	0.013
qi-mse-ng	0.1	0.4	0.141	0.003	0.028	0.772	0.007	0.022	0.011
qi-mse-ng	0.1	0.5	0.14	0.003	0.026	0.772	0.006	0.022	0.01
qi-mse-ng	0.1	0.6	0.14	0.002	0.025	0.772	0.006	0.022	0.01
qi-mse-ng	0.1	0.7	0.14	0.002	0.024	0.773	0.006	0.023	0.01
qi-mse-ng	0.1	0.8	0.141	0.002	0.025	0.776	0.008	0.026	0.011
qi-mse-ng	0.1	0.9	0.146	0.003	0.03	0.781	0.013	0.034	0.015
qi-mse-ng	0.3	—	0.511	0.052	0.102	1.024	0.57	0.82	0.5
qi-mse-ng	0.3	0.1	0.155	0.013	0.049	0.781	0.026	0.035	0.025
qi-mse-ng	0.3	0.2	0.148	0.007	0.039	0.776	0.019	0.028	0.018
qi-mse-ng	0.3	0.3	0.145	0.005	0.034	0.775	0.016	0.026	0.015
qi-mse-ng	0.3	0.4	0.144	0.004	0.032	0.774	0.014	0.025	0.014
qi-mse-ng	0.3	0.5	0.143	0.004	0.03	0.775	0.013	0.025	0.014
qi-mse-ng	0.3	0.6	0.144	0.004	0.029	0.775	0.013	0.027	0.014
qi-mse-ng	0.3	0.7	0.145	0.004	0.031	0.778	0.014	0.029	0.014
qi-mse-ng	0.3	0.8	0.148	0.004	0.036	0.783	0.016	0.034	0.015
qi-mse-ng	0.3	0.9	0.156	0.004	0.051	0.793	0.022	0.046	0.02
qi-mse-ng	0.5	—	0.976	0.207	0.492	1.644	1.005	1.589	0.918
qi-mse-ng	0.5	0.1	0.351	0.11	0.238	1.222	0.073	0.226	0.236
qi-mse-ng	0.5	0.2	0.349	0.105	0.231	1.22	0.068	0.23	0.238
qi-mse-ng	0.5	0.3	0.343	0.104	0.23	1.192	0.065	0.228	0.237
qi-mse-ng	0.5	0.4	0.348	0.099	0.229	1.225	0.064	0.231	0.236
qi-mse-ng	0.5	0.5	0.35	0.099	0.233	1.229	0.064	0.238	0.237
qi-mse-ng	0.5	0.6	0.352	0.104	0.236	1.226	0.065	0.243	0.238
qi-mse-ng	0.5	0.7	0.357	0.105	0.244	1.235	0.067	0.251	0.241
qi-mse-ng	0.5	0.8	0.359	0.105	0.255	1.21	0.07	0.262	0.249
qi-mse-ng	0.5	0.9	0.372	0.109	0.28	1.225	0.079	0.282	0.255
qi-mse-ng	0.7	—	1.61	0.69	1.151	2.285	1.408	2.483	1.641
qi-mse-ng	0.7	0.1	0.725	0.472	0.781	1.767	0.216	0.547	0.564
qi-mse-ng	0.7	0.2	0.723	0.459	0.777	1.765	0.22	0.552	0.562
qi-mse-ng	0.7	0.3	0.722	0.458	0.778	1.768	0.212	0.554	0.563
qi-mse-ng	0.7	0.4	0.727	0.466	0.783	1.77	0.224	0.558	0.563
qi-mse-ng	0.7	0.5	0.731	0.47	0.787	1.775	0.228	0.563	0.564
qi-mse-ng	0.7	0.6	0.735	0.47	0.792	1.778	0.23	0.571	0.567
qi-mse-ng	0.7	0.7	0.741	0.471	0.799	1.784	0.233	0.589	0.57
qi-mse-ng	0.7	0.8	0.753	0.473	0.819	1.791	0.244	0.612	0.578
qi-mse-ng	0.7	0.9	0.77	0.477	0.847	1.809	0.258	0.641	0.586
qi-mse-ng	0.9	—	2.906	3.38	2.259	3.461	2.235	3.448	2.651
qi-mse-ng	0.9	0.1	1.918	3.022	1.784	2.911	0.932	1.384	1.477
qi-mse-ng	0.9	0.2	1.925	3.037	1.79	2.913	0.937	1.39	1.48
qi-mse-ng	0.9	0.3	1.93	3.042	1.797	2.916	0.941	1.402	1.483
qi-mse-ng	0.9	0.4	1.937	3.056	1.8	2.923	0.945	1.412	1.488
qi-mse-ng	0.9	0.5	1.947	3.063	1.826	2.927	0.948	1.423	1.493
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 19 (VOI_s on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse-ng	0.9	0.6	1.953	3.071	1.84	2.931	0.952	1.431	1.495
qi-mse-ng	0.9	0.7	1.967	3.086	1.851	2.942	0.958	1.457	1.511
qi-mse-ng	0.9	0.8	1.979	3.097	1.867	2.949	0.967	1.469	1.523
qi-mse-ng	0.9	0.9	2.002	3.12	1.902	2.965	0.985	1.502	1.539
qi-mls-pre	0.1	—	1.079	0.973	0.842	1.851	0.689	1.194	0.925
qi-mls-pre	0.1	0.1	0.804	0.871	0.684	1.599	0.533	0.658	0.476
qi-mls-pre	0.1	0.2	0.808	0.863	0.672	1.599	0.577	0.665	0.475
qi-mls-pre	0.1	0.3	0.811	0.86	0.662	1.602	0.575	0.682	0.485
qi-mls-pre	0.1	0.4	0.814	0.861	0.657	1.603	0.583	0.689	0.488
qi-mls-pre	0.1	0.5	0.825	0.86	0.658	1.608	0.599	0.731	0.496
qi-mls-pre	0.1	0.6	0.803	0.859	0.657	1.611	0.441	0.75	0.503
qi-mls-pre	0.1	0.7	0.811	0.859	0.66	1.615	0.454	0.767	0.512
qi-mls-pre	0.1	0.8	0.832	0.86	0.668	1.624	0.481	0.826	0.53
qi-mls-pre	0.1	0.9	0.878	0.866	0.69	1.64	0.564	0.94	0.566
qi-mls-pre	0.3	—	6.513	6.645	6.267	6.731	6.424	6.275	6.735
qi-mls-pre	0.3	0.1	5.587	6.403	5.842	6.211	4.945	4.426	5.696
qi-mls-pre	0.3	0.2	5.596	6.401	5.843	6.215	4.96	4.45	5.705
qi-mls-pre	0.3	0.3	5.604	6.4	5.844	6.219	4.974	4.472	5.713
qi-mls-pre	0.3	0.4	5.612	6.399	5.845	6.222	4.987	4.496	5.721
qi-mls-pre	0.3	0.5	5.622	6.399	5.846	6.226	5.005	4.524	5.732
qi-mls-pre	0.3	0.6	5.644	6.399	5.851	6.232	5.078	4.561	5.744
qi-mls-pre	0.3	0.7	5.66	6.399	5.858	6.239	5.101	4.606	5.757
qi-mls-pre	0.3	0.8	5.686	6.399	5.868	6.248	5.146	4.674	5.778
qi-mls-pre	0.3	0.9	5.738	6.4	5.889	6.267	5.23	4.818	5.826
qi-mls-pre	0.5	—	6.784	6.793	6.578	6.932	6.694	6.747	6.959
qi-mls-pre	0.5	0.1	5.741	6.511	6.024	6.366	5.058	4.648	5.837
qi-mls-pre	0.5	0.2	5.748	6.508	6.025	6.369	5.072	4.67	5.844
qi-mls-pre	0.5	0.3	5.755	6.506	6.026	6.372	5.085	4.691	5.852
qi-mls-pre	0.5	0.4	5.763	6.505	6.027	6.375	5.097	4.716	5.859
qi-mls-pre	0.5	0.5	5.774	6.505	6.029	6.38	5.115	4.743	5.87
qi-mls-pre	0.5	0.6	5.795	6.505	6.033	6.385	5.188	4.779	5.881
qi-mls-pre	0.5	0.7	5.811	6.505	6.04	6.392	5.211	4.824	5.894
qi-mls-pre	0.5	0.8	5.837	6.505	6.05	6.401	5.255	4.893	5.915
qi-mls-pre	0.5	0.9	5.889	6.505	6.071	6.42	5.339	5.035	5.962
qi-mls-pre	0.7	—	6.784	6.793	6.578	6.932	6.694	6.747	6.959
qi-mls-pre	0.7	0.1	5.741	6.511	6.024	6.366	5.058	4.648	5.837
qi-mls-pre	0.7	0.2	5.748	6.508	6.025	6.369	5.072	4.67	5.844
qi-mls-pre	0.7	0.3	5.755	6.506	6.026	6.372	5.085	4.691	5.852
qi-mls-pre	0.7	0.4	5.763	6.505	6.027	6.375	5.097	4.716	5.859
qi-mls-pre	0.7	0.5	5.774	6.505	6.029	6.38	5.115	4.743	5.87
qi-mls-pre	0.7	0.6	5.795	6.505	6.033	6.385	5.188	4.779	5.881
qi-mls-pre	0.7	0.7	5.811	6.505	6.04	6.392	5.211	4.824	5.894
qi-mls-pre	0.7	0.8	5.837	6.505	6.05	6.401	5.255	4.893	5.915
qi-mls-pre	0.7	0.9	5.889	6.505	6.071	6.42	5.339	5.035	5.962
qi-mls-pre	0.9	—	6.784	6.793	6.578	6.932	6.694	6.747	6.959
qi-mls-pre	0.9	0.1	5.741	6.511	6.024	6.366	5.058	4.648	5.837
qi-mls-pre	0.9	0.2	5.748	6.508	6.025	6.369	5.072	4.67	5.844
qi-mls-pre	0.9	0.3	5.755	6.506	6.026	6.372	5.085	4.691	5.852
qi-mls-pre	0.9	0.4	5.763	6.505	6.027	6.375	5.097	4.716	5.859
qi-mls-pre	0.9	0.5	5.774	6.505	6.029	6.38	5.115	4.743	5.87
qi-mls-pre	0.9	0.6	5.795	6.505	6.033	6.385	5.188	4.779	5.881
qi-mls-pre	0.9	0.7	5.811	6.505	6.04	6.392	5.211	4.824	5.894
qi-mls-pre	0.9	0.8	5.837	6.505	6.05	6.401	5.255	4.893	5.915
qi-mls-pre	0.9	0.9	5.889	6.505	6.071	6.42	5.339	5.035	5.962
qi-mls-pre-ng	0.1	—	0.753	0.52	0.536	1.028	0.973	0.898	0.563
qi-mls-pre-ng	0.1	0.1	0.56	0.423	0.534	0.732	0.567	0.612	0.494
qi-mls-pre-ng	0.1	0.2	0.549	0.423	0.536	0.765	0.56	0.519	0.49
qi-mls-pre-ng	0.1	0.3	0.59	0.42	0.533	0.918	0.564	0.614	0.489
qi-mls-pre-ng	0.1	0.4	0.552	0.43	0.529	0.763	0.551	0.55	0.487
qi-mls-pre-ng	0.1	0.5	0.558	0.425	0.537	0.763	0.586	0.55	0.486
qi-mls-pre-ng	0.1	0.6	0.562	0.412	0.568	0.766	0.586	0.552	0.487
qi-mls-pre-ng	0.1	0.7	0.578	0.413	0.577	0.767	0.67	0.551	0.487
qi-mls-pre-ng	0.1	0.8	0.587	0.419	0.587	0.816	0.658	0.566	0.476
qi-mls-pre-ng	0.1	0.9	0.602	0.442	0.588	0.824	0.638	0.593	0.527
qi-mls-pre-ng	0.3	—	2.851	2.535	2.913	3.002	2.695	3.046	2.914
qi-mls-pre-ng	0.3	0.1	2.217	2.398	2.74	2.75	1.489	1.84	2.082
qi-mls-pre-ng	0.3	0.2	2.214	2.393	2.738	2.746	1.489	1.839	2.08
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 19 (VOI_s on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre-ng	0.3	0.3	2.213	2.391	2.735	2.747	1.488	1.838	2.081
qi-mls-pre-ng	0.3	0.4	2.216	2.392	2.733	2.748	1.488	1.857	2.082
qi-mls-pre-ng	0.3	0.5	2.223	2.392	2.734	2.748	1.523	1.86	2.083
qi-mls-pre-ng	0.3	0.6	2.226	2.392	2.74	2.751	1.523	1.863	2.083
qi-mls-pre-ng	0.3	0.7	2.254	2.392	2.744	2.752	1.682	1.869	2.084
qi-mls-pre-ng	0.3	0.8	2.26	2.393	2.752	2.762	1.687	1.881	2.088
qi-mls-pre-ng	0.3	0.9	2.324	2.398	2.775	2.772	1.943	1.91	2.145
qi-mls-pre-ng	0.5	—	2.851	2.535	2.913	3.002	2.696	3.046	2.916
qi-mls-pre-ng	0.5	0.1	2.217	2.398	2.74	2.75	1.489	1.84	2.083
qi-mls-pre-ng	0.5	0.2	2.214	2.393	2.738	2.746	1.489	1.839	2.081
qi-mls-pre-ng	0.5	0.3	2.214	2.391	2.735	2.747	1.489	1.838	2.082
qi-mls-pre-ng	0.5	0.4	2.217	2.392	2.733	2.748	1.488	1.857	2.083
qi-mls-pre-ng	0.5	0.5	2.223	2.392	2.734	2.748	1.523	1.86	2.084
qi-mls-pre-ng	0.5	0.6	2.226	2.392	2.74	2.751	1.524	1.863	2.084
qi-mls-pre-ng	0.5	0.7	2.254	2.392	2.744	2.752	1.682	1.869	2.085
qi-mls-pre-ng	0.5	0.8	2.261	2.393	2.752	2.762	1.688	1.881	2.089
qi-mls-pre-ng	0.5	0.9	2.324	2.398	2.775	2.772	1.944	1.91	2.146
qi-mls-pre-ng	0.7	—	2.851	2.535	2.913	3.002	2.696	3.046	2.916
qi-mls-pre-ng	0.7	0.1	2.217	2.398	2.74	2.75	1.489	1.84	2.083
qi-mls-pre-ng	0.7	0.2	2.214	2.393	2.738	2.746	1.489	1.839	2.081
qi-mls-pre-ng	0.7	0.3	2.214	2.391	2.735	2.747	1.489	1.838	2.082
qi-mls-pre-ng	0.7	0.4	2.217	2.392	2.733	2.748	1.488	1.857	2.083
qi-mls-pre-ng	0.7	0.5	2.223	2.392	2.734	2.748	1.523	1.86	2.084
qi-mls-pre-ng	0.7	0.6	2.226	2.392	2.74	2.751	1.524	1.863	2.084
qi-mls-pre-ng	0.7	0.7	2.254	2.392	2.744	2.752	1.682	1.869	2.085
qi-mls-pre-ng	0.7	0.8	2.261	2.393	2.752	2.762	1.688	1.881	2.089
qi-mls-pre-ng	0.7	0.9	2.324	2.398	2.775	2.772	1.944	1.91	2.146
qi-mls-pre-ng	0.9	—	2.851	2.535	2.913	3.002	2.696	3.046	2.916
qi-mls-pre-ng	0.9	0.1	2.217	2.398	2.74	2.75	1.489	1.84	2.083
qi-mls-pre-ng	0.9	0.2	2.214	2.393	2.738	2.746	1.489	1.839	2.081
qi-mls-pre-ng	0.9	0.3	2.214	2.391	2.735	2.747	1.489	1.838	2.082
qi-mls-pre-ng	0.9	0.4	2.217	2.392	2.733	2.748	1.488	1.857	2.083
qi-mls-pre-ng	0.9	0.5	2.223	2.392	2.734	2.748	1.523	1.86	2.084
qi-mls-pre-ng	0.9	0.6	2.226	2.392	2.74	2.751	1.524	1.863	2.084
qi-mls-pre-ng	0.9	0.7	2.254	2.392	2.744	2.752	1.682	1.869	2.085
qi-mls-pre-ng	0.9	0.8	2.261	2.393	2.752	2.762	1.688	1.881	2.089
qi-mls-pre-ng	0.9	0.9	2.324	2.398	2.775	2.772	1.944	1.91	2.146
qi-mls	0.1	—	1.479	1.067	1.268	2.292	1.087	1.705	1.457
qi-mls	0.1	0.1	0.988	0.956	1.068	2.039	0.39	0.713	0.763
qi-mls	0.1	0.2	0.984	0.954	1.063	2.04	0.381	0.708	0.759
qi-mls	0.1	0.3	0.984	0.956	1.062	2.04	0.377	0.711	0.756
qi-mls	0.1	0.4	0.985	0.962	1.061	2.037	0.38	0.716	0.753
qi-mls	0.1	0.5	0.99	0.962	1.062	2.042	0.382	0.735	0.754
qi-mls	0.1	0.6	0.992	0.962	1.067	2.046	0.378	0.74	0.757
qi-mls	0.1	0.7	0.996	0.966	1.069	2.049	0.38	0.751	0.76
qi-mls	0.1	0.8	1.005	0.972	1.077	2.054	0.385	0.777	0.763
qi-mls	0.1	0.9	1.021	0.982	1.1	2.066	0.399	0.807	0.774
qi-mls	0.3	—	3.614	3.38	3.605	4.284	3.224	3.688	3.504
qi-mls	0.3	0.1	3.012	3.198	3.386	4.0	2.32	2.407	2.761
qi-mls	0.3	0.2	3.027	3.207	3.399	4.01	2.34	2.438	2.77
qi-mls	0.3	0.3	3.038	3.216	3.406	4.019	2.355	2.461	2.775
qi-mls	0.3	0.4	3.049	3.225	3.417	4.025	2.37	2.473	2.783
qi-mls	0.3	0.5	3.056	3.231	3.423	4.033	2.373	2.482	2.796
qi-mls	0.3	0.6	3.073	3.243	3.431	4.04	2.417	2.498	2.809
qi-mls	0.3	0.7	3.085	3.256	3.44	4.049	2.438	2.513	2.818
qi-mls	0.3	0.8	3.1	3.27	3.453	4.058	2.45	2.531	2.836
qi-mls	0.3	0.9	3.127	3.302	3.477	4.076	2.481	2.573	2.857
qi-mls	0.5	—	11.324	10.878	11.735	11.625	12.431	10.182	11.093
qi-mls	0.5	0.1	10.735	10.772	11.508	11.361	11.473	8.936	10.36
qi-mls	0.5	0.2	10.743	10.774	11.521	11.366	11.484	8.947	10.365
qi-mls	0.5	0.3	10.747	10.774	11.529	11.369	11.489	8.954	10.368
qi-mls	0.5	0.4	10.75	10.774	11.536	11.371	11.493	8.958	10.37
qi-mls	0.5	0.5	10.753	10.775	11.541	11.374	11.496	8.962	10.372
qi-mls	0.5	0.6	10.756	10.775	11.546	11.376	11.498	8.967	10.373
qi-mls	0.5	0.7	10.759	10.775	11.551	11.379	11.501	8.972	10.374
qi-mls	0.5	0.8	10.763	10.776	11.558	11.383	11.505	8.98	10.377
qi-mls	0.5	0.9	10.776	10.777	11.578	11.392	11.517	9.005	10.385
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 19 (VOI_s on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls	0.7	—	11.324	10.878	11.735	11.625	12.431	10.182	11.093
qi-mls	0.7	0.1	10.735	10.772	11.508	11.361	11.473	8.936	10.36
qi-mls	0.7	0.2	10.743	10.774	11.521	11.366	11.484	8.947	10.365
qi-mls	0.7	0.3	10.747	10.774	11.529	11.369	11.489	8.954	10.368
qi-mls	0.7	0.4	10.75	10.774	11.536	11.371	11.493	8.958	10.37
qi-mls	0.7	0.5	10.753	10.775	11.541	11.374	11.496	8.962	10.372
qi-mls	0.7	0.6	10.756	10.775	11.546	11.376	11.498	8.967	10.373
qi-mls	0.7	0.7	10.759	10.775	11.551	11.379	11.501	8.972	10.374
qi-mls	0.7	0.8	10.763	10.776	11.558	11.383	11.505	8.98	10.377
qi-mls	0.7	0.9	10.776	10.777	11.578	11.392	11.517	9.005	10.385
qi-mls	0.9	—	11.324	10.878	11.735	11.625	12.431	10.182	11.093
qi-mls	0.9	0.1	10.735	10.772	11.508	11.361	11.473	8.936	10.36
qi-mls	0.9	0.2	10.743	10.774	11.521	11.366	11.484	8.947	10.365
qi-mls	0.9	0.3	10.747	10.774	11.529	11.369	11.489	8.954	10.368
qi-mls	0.9	0.4	10.75	10.774	11.536	11.371	11.493	8.958	10.37
qi-mls	0.9	0.5	10.753	10.775	11.541	11.374	11.496	8.962	10.372
qi-mls	0.9	0.6	10.756	10.775	11.546	11.376	11.498	8.967	10.373
qi-mls	0.9	0.7	10.759	10.775	11.551	11.379	11.501	8.972	10.374
qi-mls	0.9	0.8	10.763	10.776	11.558	11.383	11.505	8.98	10.377
qi-mls	0.9	0.9	10.776	10.777	11.578	11.392	11.517	9.005	10.385
qi-mls-ng	0.1	—	0.194	0.391	0.068	0.133	0.18	0.104	0.291
qi-mls-ng	0.1	0.1	0.141	0.377	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.1	0.2	0.141	0.377	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.1	0.3	0.148	0.377	0.068	0.13	0.087	0.052	0.174
qi-mls-ng	0.1	0.4	0.153	0.377	0.068	0.13	0.116	0.052	0.174
qi-mls-ng	0.1	0.5	0.153	0.378	0.068	0.13	0.116	0.052	0.175
qi-mls-ng	0.1	0.6	0.155	0.378	0.068	0.13	0.128	0.053	0.175
qi-mls-ng	0.1	0.7	0.159	0.378	0.068	0.13	0.128	0.053	0.194
qi-mls-ng	0.1	0.8	0.159	0.378	0.068	0.13	0.129	0.054	0.195
qi-mls-ng	0.1	0.9	0.16	0.381	0.068	0.13	0.129	0.056	0.197
qi-mls-ng	0.3	—	0.195	0.397	0.068	0.133	0.18	0.104	0.291
qi-mls-ng	0.3	0.1	0.141	0.38	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.3	0.2	0.142	0.381	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.3	0.3	0.149	0.381	0.068	0.13	0.087	0.052	0.174
qi-mls-ng	0.3	0.4	0.153	0.381	0.068	0.13	0.116	0.052	0.174
qi-mls-ng	0.3	0.5	0.154	0.381	0.068	0.13	0.116	0.052	0.175
qi-mls-ng	0.3	0.6	0.156	0.381	0.068	0.13	0.128	0.053	0.175
qi-mls-ng	0.3	0.7	0.159	0.381	0.068	0.13	0.128	0.053	0.194
qi-mls-ng	0.3	0.8	0.159	0.381	0.068	0.13	0.129	0.054	0.195
qi-mls-ng	0.3	0.9	0.161	0.386	0.068	0.13	0.129	0.056	0.197
qi-mls-ng	0.5	—	0.195	0.397	0.068	0.133	0.18	0.104	0.291
qi-mls-ng	0.5	0.1	0.141	0.38	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.5	0.2	0.142	0.381	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.5	0.3	0.149	0.381	0.068	0.13	0.087	0.052	0.174
qi-mls-ng	0.5	0.4	0.153	0.381	0.068	0.13	0.116	0.052	0.174
qi-mls-ng	0.5	0.5	0.154	0.381	0.068	0.13	0.116	0.052	0.175
qi-mls-ng	0.5	0.6	0.156	0.381	0.068	0.13	0.128	0.053	0.175
qi-mls-ng	0.5	0.7	0.159	0.381	0.068	0.13	0.128	0.053	0.194
qi-mls-ng	0.5	0.8	0.159	0.381	0.068	0.13	0.129	0.054	0.195
qi-mls-ng	0.5	0.9	0.161	0.386	0.068	0.13	0.129	0.056	0.197
qi-mls-ng	0.7	—	0.195	0.397	0.068	0.133	0.18	0.104	0.291
qi-mls-ng	0.7	0.1	0.141	0.38	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.7	0.2	0.142	0.381	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.7	0.3	0.149	0.381	0.068	0.13	0.087	0.052	0.174
qi-mls-ng	0.7	0.4	0.153	0.381	0.068	0.13	0.116	0.052	0.174
qi-mls-ng	0.7	0.5	0.154	0.381	0.068	0.13	0.116	0.052	0.175
qi-mls-ng	0.7	0.6	0.156	0.381	0.068	0.13	0.128	0.053	0.175
qi-mls-ng	0.7	0.7	0.159	0.381	0.068	0.13	0.128	0.053	0.194
qi-mls-ng	0.7	0.8	0.159	0.381	0.068	0.13	0.129	0.054	0.195
qi-mls-ng	0.7	0.9	0.161	0.386	0.068	0.13	0.129	0.056	0.197
qi-mls-ng	0.9	—	0.195	0.397	0.068	0.133	0.18	0.104	0.291
qi-mls-ng	0.9	0.1	0.141	0.38	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.9	0.2	0.142	0.381	0.068	0.13	0.087	0.051	0.132
qi-mls-ng	0.9	0.3	0.149	0.381	0.068	0.13	0.087	0.052	0.174
qi-mls-ng	0.9	0.4	0.153	0.381	0.068	0.13	0.116	0.052	0.174
qi-mls-ng	0.9	0.5	0.154	0.381	0.068	0.13	0.116	0.052	0.175
qi-mls-ng	0.9	0.6	0.156	0.381	0.068	0.13	0.128	0.053	0.175
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 19 (VOI_s on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-ng	0.9	0.7	0.159	0.381	0.068	0.13	0.128	0.053	0.194
qi-mls-ng	0.9	0.8	0.159	0.381	0.068	0.13	0.129	0.054	0.195
qi-mls-ng	0.9	0.9	0.161	0.386	0.068	0.13	0.129	0.056	0.197
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Table 20: VOI_m performance on 100% of the data including training data for all architectures and parameter sets. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging.

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	—	5.969	7.706	6.604	5.916	4.365	5.397	5.824
qi-mse	0.1	0.1	5.487	7.561	6.359	5.637	3.713	4.687	4.963
qi-mse	0.1	0.2	5.534	7.558	6.356	5.633	3.719	4.68	5.258
qi-mse	0.1	0.3	5.547	7.556	6.352	5.632	3.719	4.77	5.255
qi-mse	0.1	0.4	5.556	7.556	6.355	5.631	3.714	4.769	5.31
qi-mse	0.1	0.5	5.556	7.556	6.356	5.632	3.713	4.769	5.31
qi-mse	0.1	0.6	5.557	7.557	6.357	5.633	3.712	4.77	5.31
qi-mse	0.1	0.7	5.557	7.557	6.359	5.634	3.712	4.772	5.31
qi-mse	0.1	0.8	5.557	7.557	6.363	5.636	3.698	4.775	5.311
qi-mse	0.1	0.9	5.56	7.559	6.37	5.64	3.699	4.781	5.312
qi-mse	0.3	—	5.635	7.655	6.553	5.811	3.238	5.183	5.372
qi-mse	0.3	0.1	5.309	7.553	6.331	5.576	2.786	4.663	4.947
qi-mse	0.3	0.2	5.309	7.55	6.325	5.613	2.776	4.652	4.939
qi-mse	0.3	0.3	5.293	7.51	6.321	5.569	2.771	4.649	4.935
qi-mse	0.3	0.4	5.291	7.51	6.319	5.569	2.769	4.647	4.933
qi-mse	0.3	0.5	5.29	7.51	6.317	5.569	2.767	4.646	4.931
qi-mse	0.3	0.6	5.286	7.51	6.315	5.567	2.749	4.646	4.931
qi-mse	0.3	0.7	5.286	7.51	6.315	5.566	2.749	4.644	4.93
qi-mse	0.3	0.8	5.285	7.51	6.314	5.566	2.749	4.644	4.93
qi-mse	0.3	0.9	5.286	7.51	6.315	5.568	2.748	4.642	4.931
qi-mse	0.5	—	1.658	1.465	2.868	0.911	0.908	2.13	1.664
qi-mse	0.5	0.1	1.561	1.597	2.891	1.114	0.607	1.556	1.601
qi-mse	0.5	0.2	1.5	1.478	2.79	1.067	0.585	1.535	1.545
qi-mse	0.5	0.3	1.501	1.529	2.762	1.033	0.566	1.627	1.486
qi-mse	0.5	0.4	1.499	1.482	2.734	1.022	0.569	1.614	1.574
qi-mse	0.5	0.5	1.462	1.507	2.722	1.031	0.546	1.511	1.453
qi-mse	0.5	0.6	1.415	1.441	2.516	1.017	0.536	1.53	1.452
qi-mse	0.5	0.7	1.416	1.451	2.509	1.024	0.535	1.527	1.447
qi-mse	0.5	0.8	1.419	1.446	2.506	1.009	0.534	1.567	1.454
qi-mse	0.5	0.9	1.359	1.451	2.503	1.009	0.504	1.475	1.212
qi-mse	0.7	—	0.237	0.085	0.203	0.137	0.386	0.305	0.306
qi-mse	0.7	0.1	0.116	0.037	0.294	0.088	0.106	0.09	0.079
qi-mse	0.7	0.2	0.094	0.024	0.27	0.076	0.065	0.068	0.064
qi-mse	0.7	0.3	0.087	0.02	0.255	0.071	0.058	0.059	0.057
qi-mse	0.7	0.4	0.081	0.019	0.242	0.065	0.054	0.054	0.053
qi-mse	0.7	0.5	0.079	0.018	0.235	0.067	0.052	0.051	0.051
qi-mse	0.7	0.6	0.076	0.018	0.227	0.066	0.05	0.043	0.05
qi-mse	0.7	0.7	0.074	0.018	0.222	0.066	0.048	0.041	0.049
qi-mse	0.7	0.8	0.073	0.018	0.219	0.065	0.047	0.04	0.049
qi-mse	0.7	0.9	0.073	0.018	0.217	0.066	0.048	0.04	0.049
qi-mse	0.9	—	0.17	0.058	0.079	0.069	0.333	0.239	0.241
qi-mse	0.9	0.1	0.069	0.024	0.142	0.036	0.076	0.066	0.07
qi-mse	0.9	0.2	0.051	0.011	0.114	0.023	0.06	0.045	0.054
qi-mse	0.9	0.3	0.043	0.007	0.099	0.018	0.053	0.035	0.048
qi-mse	0.9	0.4	0.039	0.006	0.089	0.016	0.049	0.03	0.044
qi-mse	0.9	0.5	0.036	0.005	0.082	0.015	0.047	0.027	0.042
qi-mse	0.9	0.6	0.034	0.005	0.077	0.014	0.045	0.024	0.041
qi-mse	0.9	0.7	0.033	0.005	0.072	0.013	0.044	0.022	0.04
qi-mse	0.9	0.8	0.032	0.005	0.069	0.013	0.043	0.021	0.039
qi-mse	0.9	0.9	0.031	0.005	0.066	0.013	0.044	0.021	0.039
qi-mse-ng	0.1	—	6.038	7.705	6.604	5.917	4.688	5.494	5.82
qi-mse-ng	0.1	0.1	5.496	7.519	6.278	5.621	3.517	4.779	5.261
qi-mse-ng	0.1	0.2	5.53	7.517	6.353	5.62	3.659	4.774	5.256
qi-mse-ng	0.1	0.3	5.537	7.555	6.353	5.619	3.671	4.773	5.254
qi-mse-ng	0.1	0.4	5.537	7.555	6.354	5.62	3.67	4.772	5.253
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 20 (VOI_m on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse-ng	0.1	0.5	5.544	7.554	6.355	5.643	3.69	4.772	5.252
qi-mse-ng	0.1	0.6	5.545	7.554	6.357	5.644	3.689	4.773	5.251
qi-mse-ng	0.1	0.7	5.547	7.554	6.362	5.646	3.689	4.776	5.251
qi-mse-ng	0.1	0.8	5.549	7.555	6.367	5.649	3.69	4.78	5.252
qi-mse-ng	0.1	0.9	5.549	7.555	6.378	5.631	3.69	4.787	5.254
qi-mse-ng	0.3	—	5.882	7.64	6.556	5.831	4.21	5.352	5.704
qi-mse-ng	0.3	0.1	5.493	7.518	6.274	5.618	3.515	4.775	5.259
qi-mse-ng	0.3	0.2	5.526	7.516	6.345	5.616	3.654	4.769	5.253
qi-mse-ng	0.3	0.3	5.524	7.515	6.346	5.615	3.651	4.767	5.251
qi-mse-ng	0.3	0.4	5.523	7.514	6.346	5.615	3.649	4.766	5.249
qi-mse-ng	0.3	0.5	5.473	7.514	6.348	5.616	3.647	4.766	4.945
qi-mse-ng	0.3	0.6	5.473	7.513	6.349	5.617	3.647	4.766	4.944
qi-mse-ng	0.3	0.7	5.473	7.513	6.35	5.616	3.646	4.767	4.943
qi-mse-ng	0.3	0.8	5.444	7.513	6.353	5.438	3.646	4.768	4.943
qi-mse-ng	0.3	0.9	5.446	7.513	6.354	5.439	3.649	4.773	4.945
qi-mse-ng	0.5	—	4.099	5.293	5.56	3.006	2.513	3.968	4.256
qi-mse-ng	0.5	0.1	3.927	5.197	5.565	2.937	2.343	3.581	3.941
qi-mse-ng	0.5	0.2	3.837	5.193	5.538	2.849	2.045	3.501	3.898
qi-mse-ng	0.5	0.3	3.836	5.073	5.526	3.001	2.041	3.503	3.871
qi-mse-ng	0.5	0.4	3.807	5.061	5.528	2.866	2.025	3.472	3.892
qi-mse-ng	0.5	0.5	3.757	5.04	5.519	2.81	2.013	3.597	3.562
qi-mse-ng	0.5	0.6	3.725	5.003	5.515	2.814	1.997	3.478	3.543
qi-mse-ng	0.5	0.7	3.7	4.974	5.485	2.732	1.979	3.474	3.556
qi-mse-ng	0.5	0.8	3.715	4.957	5.483	2.88	1.978	3.471	3.519
qi-mse-ng	0.5	0.9	3.711	4.948	5.478	2.878	1.981	3.47	3.515
qi-mse-ng	0.7	—	0.313	0.139	0.381	0.176	0.47	0.369	0.343
qi-mse-ng	0.7	0.1	0.149	0.068	0.418	0.128	0.066	0.122	0.091
qi-mse-ng	0.7	0.2	0.137	0.062	0.4	0.119	0.056	0.106	0.079
qi-mse-ng	0.7	0.3	0.131	0.059	0.391	0.115	0.052	0.094	0.074
qi-mse-ng	0.7	0.4	0.126	0.057	0.379	0.112	0.049	0.089	0.071
qi-mse-ng	0.7	0.5	0.124	0.056	0.373	0.11	0.047	0.086	0.069
qi-mse-ng	0.7	0.6	0.122	0.056	0.368	0.109	0.046	0.084	0.067
qi-mse-ng	0.7	0.7	0.12	0.056	0.364	0.108	0.045	0.082	0.066
qi-mse-ng	0.7	0.8	0.113	0.056	0.326	0.107	0.045	0.081	0.066
qi-mse-ng	0.7	0.9	0.114	0.056	0.322	0.107	0.048	0.083	0.068
qi-mse-ng	0.9	—	0.188	0.079	0.097	0.069	0.371	0.26	0.25
qi-mse-ng	0.9	0.1	0.058	0.025	0.134	0.034	0.027	0.067	0.061
qi-mse-ng	0.9	0.2	0.046	0.017	0.116	0.024	0.016	0.051	0.049
qi-mse-ng	0.9	0.3	0.04	0.015	0.107	0.02	0.012	0.044	0.044
qi-mse-ng	0.9	0.4	0.037	0.014	0.1	0.017	0.009	0.039	0.041
qi-mse-ng	0.9	0.5	0.034	0.013	0.094	0.016	0.007	0.036	0.039
qi-mse-ng	0.9	0.6	0.032	0.013	0.089	0.014	0.006	0.034	0.037
qi-mse-ng	0.9	0.7	0.031	0.013	0.085	0.013	0.005	0.032	0.036
qi-mse-ng	0.9	0.8	0.03	0.012	0.082	0.013	0.005	0.031	0.036
qi-mse-ng	0.9	0.9	0.031	0.012	0.08	0.012	0.008	0.033	0.037
qi-mls-pre	0.1	—	3.559	3.607	4.102	3.016	3.341	3.143	4.144
qi-mls-pre	0.1	0.1	3.266	4.023	4.233	2.911	1.844	3.023	3.565
qi-mls-pre	0.1	0.2	3.247	4.025	4.223	2.902	1.766	3.017	3.55
qi-mls-pre	0.1	0.3	3.224	4.022	4.22	2.897	1.765	2.971	3.469
qi-mls-pre	0.1	0.4	3.231	4.02	4.212	2.878	1.792	3.019	3.467
qi-mls-pre	0.1	0.5	3.221	4.019	4.177	2.876	1.919	2.881	3.452
qi-mls-pre	0.1	0.6	3.225	4.03	4.172	2.875	1.924	2.894	3.452
qi-mls-pre	0.1	0.7	3.231	4.027	4.167	2.875	1.955	2.912	3.451
qi-mls-pre	0.1	0.8	3.253	4.026	4.135	2.875	2.136	2.89	3.458
qi-mls-pre	0.1	0.9	3.251	4.027	4.118	2.872	2.132	2.889	3.468
qi-mls-pre	0.3	—	0.985	1.071	1.157	1.263	0.546	0.924	0.951
qi-mls-pre	0.3	0.1	0.961	1.024	1.195	1.2	0.656	0.826	0.867
qi-mls-pre	0.3	0.2	0.943	1.014	1.169	1.188	0.64	0.799	0.85
qi-mls-pre	0.3	0.3	0.934	1.011	1.151	1.182	0.631	0.785	0.846
qi-mls-pre	0.3	0.4	0.928	1.009	1.136	1.179	0.626	0.775	0.841
qi-mls-pre	0.3	0.5	0.924	1.008	1.126	1.177	0.623	0.77	0.839
qi-mls-pre	0.3	0.6	0.892	1.008	1.118	1.175	0.45	0.766	0.836
qi-mls-pre	0.3	0.7	0.89	1.007	1.111	1.175	0.449	0.763	0.834
qi-mls-pre	0.3	0.8	0.888	1.007	1.105	1.175	0.445	0.762	0.833
qi-mls-pre	0.3	0.9	0.885	1.007	1.099	1.176	0.446	0.753	0.83
qi-mls-pre	0.5	—	0.983	1.071	1.154	1.258	0.545	0.919	0.948
qi-mls-pre	0.5	0.1	0.959	1.024	1.193	1.195	0.656	0.825	0.862
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 20 (VOI_m on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre	0.5	0.2	0.941	1.014	1.167	1.183	0.64	0.798	0.845
qi-mls-pre	0.5	0.3	0.932	1.011	1.149	1.177	0.631	0.784	0.841
qi-mls-pre	0.5	0.4	0.926	1.009	1.135	1.174	0.626	0.774	0.836
qi-mls-pre	0.5	0.5	0.922	1.008	1.124	1.172	0.623	0.769	0.834
qi-mls-pre	0.5	0.6	0.89	1.008	1.116	1.171	0.45	0.765	0.831
qi-mls-pre	0.5	0.7	0.888	1.007	1.109	1.17	0.448	0.763	0.83
qi-mls-pre	0.5	0.8	0.886	1.007	1.103	1.17	0.445	0.761	0.828
qi-mls-pre	0.5	0.9	0.883	1.007	1.097	1.171	0.446	0.752	0.825
qi-mls-pre	0.7	—	0.983	1.071	1.154	1.258	0.545	0.919	0.948
qi-mls-pre	0.7	0.1	0.959	1.024	1.193	1.195	0.656	0.825	0.862
qi-mls-pre	0.7	0.2	0.941	1.014	1.167	1.183	0.64	0.798	0.845
qi-mls-pre	0.7	0.3	0.932	1.011	1.149	1.177	0.631	0.784	0.841
qi-mls-pre	0.7	0.4	0.926	1.009	1.135	1.174	0.626	0.774	0.836
qi-mls-pre	0.7	0.5	0.922	1.008	1.124	1.172	0.623	0.769	0.834
qi-mls-pre	0.7	0.6	0.89	1.008	1.116	1.171	0.45	0.765	0.831
qi-mls-pre	0.7	0.7	0.888	1.007	1.109	1.17	0.448	0.763	0.83
qi-mls-pre	0.7	0.8	0.886	1.007	1.103	1.17	0.445	0.761	0.828
qi-mls-pre	0.7	0.9	0.883	1.007	1.097	1.171	0.446	0.752	0.825
qi-mls-pre	0.9	—	0.983	1.071	1.154	1.258	0.545	0.919	0.948
qi-mls-pre	0.9	0.1	0.959	1.024	1.193	1.195	0.656	0.825	0.862
qi-mls-pre	0.9	0.2	0.941	1.014	1.167	1.183	0.64	0.798	0.845
qi-mls-pre	0.9	0.3	0.932	1.011	1.149	1.177	0.631	0.784	0.841
qi-mls-pre	0.9	0.4	0.926	1.009	1.135	1.174	0.626	0.774	0.836
qi-mls-pre	0.9	0.5	0.922	1.008	1.124	1.172	0.623	0.769	0.834
qi-mls-pre	0.9	0.6	0.89	1.008	1.116	1.171	0.45	0.765	0.831
qi-mls-pre	0.9	0.7	0.888	1.007	1.109	1.17	0.448	0.763	0.83
qi-mls-pre	0.9	0.8	0.886	1.007	1.103	1.17	0.445	0.761	0.828
qi-mls-pre	0.9	0.9	0.883	1.007	1.097	1.171	0.446	0.752	0.825
qi-mls-pre-ng	0.1	—	5.219	6.723	5.763	5.094	3.554	4.635	5.544
qi-mls-pre-ng	0.1	0.1	4.881	6.771	5.58	5.021	3.249	3.65	5.015
qi-mls-pre-ng	0.1	0.2	4.927	6.727	5.56	4.973	3.255	4.05	4.994
qi-mls-pre-ng	0.1	0.3	4.856	6.727	5.517	4.852	3.244	3.81	4.986
qi-mls-pre-ng	0.1	0.4	4.892	6.707	5.512	4.968	3.228	3.954	4.984
qi-mls-pre-ng	0.1	0.5	4.888	6.715	5.499	4.967	3.205	3.955	4.983
qi-mls-pre-ng	0.1	0.6	4.886	6.736	5.472	4.968	3.205	3.955	4.983
qi-mls-pre-ng	0.1	0.7	4.895	6.736	5.467	4.969	3.258	3.956	4.983
qi-mls-pre-ng	0.1	0.8	4.881	6.725	5.468	4.945	3.243	3.933	4.975
qi-mls-pre-ng	0.1	0.9	4.978	6.663	5.537	4.949	3.765	3.883	5.072
qi-mls-pre-ng	0.3	—	2.274	2.483	1.945	2.796	1.924	2.151	2.346
qi-mls-pre-ng	0.3	0.1	2.226	2.374	1.931	2.714	1.929	1.944	2.467
qi-mls-pre-ng	0.3	0.2	2.203	2.368	1.9	2.701	1.91	1.894	2.448
qi-mls-pre-ng	0.3	0.3	2.197	2.371	1.888	2.704	1.887	1.886	2.445
qi-mls-pre-ng	0.3	0.4	2.184	2.373	1.88	2.707	1.859	1.842	2.442
qi-mls-pre-ng	0.3	0.5	2.181	2.373	1.878	2.706	1.849	1.841	2.441
qi-mls-pre-ng	0.3	0.6	2.179	2.373	1.869	2.706	1.849	1.838	2.44
qi-mls-pre-ng	0.3	0.7	2.175	2.373	1.867	2.707	1.829	1.835	2.44
qi-mls-pre-ng	0.3	0.8	2.166	2.374	1.867	2.709	1.802	1.827	2.419
qi-mls-pre-ng	0.3	0.9	2.127	2.371	1.86	2.704	1.7	1.765	2.362
qi-mls-pre-ng	0.5	—	2.274	2.483	1.945	2.796	1.924	2.151	2.346
qi-mls-pre-ng	0.5	0.1	2.226	2.374	1.931	2.714	1.929	1.944	2.467
qi-mls-pre-ng	0.5	0.2	2.203	2.368	1.9	2.701	1.91	1.894	2.448
qi-mls-pre-ng	0.5	0.3	2.197	2.371	1.888	2.704	1.887	1.886	2.445
qi-mls-pre-ng	0.5	0.4	2.184	2.373	1.88	2.707	1.859	1.842	2.442
qi-mls-pre-ng	0.5	0.5	2.181	2.373	1.878	2.706	1.849	1.841	2.441
qi-mls-pre-ng	0.5	0.6	2.179	2.373	1.869	2.706	1.849	1.838	2.44
qi-mls-pre-ng	0.5	0.7	2.175	2.373	1.867	2.707	1.829	1.835	2.44
qi-mls-pre-ng	0.5	0.8	2.166	2.374	1.867	2.709	1.802	1.827	2.419
qi-mls-pre-ng	0.5	0.9	2.127	2.371	1.86	2.704	1.7	1.765	2.362
qi-mls-pre-ng	0.7	—	2.274	2.483	1.945	2.796	1.924	2.151	2.346
qi-mls-pre-ng	0.7	0.1	2.226	2.374	1.931	2.714	1.929	1.944	2.467
qi-mls-pre-ng	0.7	0.2	2.203	2.368	1.9	2.701	1.91	1.894	2.448
qi-mls-pre-ng	0.7	0.3	2.197	2.371	1.888	2.704	1.887	1.886	2.445
qi-mls-pre-ng	0.7	0.4	2.184	2.373	1.88	2.707	1.859	1.842	2.442
qi-mls-pre-ng	0.7	0.5	2.181	2.373	1.878	2.706	1.849	1.841	2.441
qi-mls-pre-ng	0.7	0.6	2.179	2.373	1.869	2.706	1.849	1.838	2.44
qi-mls-pre-ng	0.7	0.7	2.175	2.373	1.867	2.707	1.829	1.835	2.44
qi-mls-pre-ng	0.7	0.8	2.166	2.374	1.867	2.709	1.802	1.827	2.419
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 20 (VOI_m on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre-ng	0.7	0.9	2.127	2.371	1.86	2.704	1.7	1.765	2.362
qi-mls-pre-ng	0.9	—	2.274	2.483	1.945	2.796	1.924	2.151	2.346
qi-mls-pre-ng	0.9	0.1	2.226	2.374	1.931	2.714	1.929	1.944	2.467
qi-mls-pre-ng	0.9	0.2	2.203	2.368	1.9	2.701	1.91	1.894	2.448
qi-mls-pre-ng	0.9	0.3	2.197	2.371	1.888	2.704	1.887	1.886	2.445
qi-mls-pre-ng	0.9	0.4	2.184	2.373	1.88	2.707	1.859	1.842	2.442
qi-mls-pre-ng	0.9	0.5	2.181	2.373	1.878	2.706	1.849	1.841	2.441
qi-mls-pre-ng	0.9	0.6	2.179	2.373	1.869	2.706	1.849	1.838	2.44
qi-mls-pre-ng	0.9	0.7	2.175	2.373	1.867	2.707	1.829	1.835	2.44
qi-mls-pre-ng	0.9	0.8	2.166	2.374	1.867	2.709	1.802	1.827	2.419
qi-mls-pre-ng	0.9	0.9	2.127	2.371	1.86	2.704	1.7	1.765	2.362
qi-mls	0.1	—	0.452	0.164	0.382	0.372	0.766	0.58	0.446
qi-mls	0.1	0.1	0.263	0.107	0.354	0.327	0.312	0.247	0.232
qi-mls	0.1	0.2	0.246	0.097	0.332	0.316	0.294	0.222	0.215
qi-mls	0.1	0.3	0.238	0.094	0.328	0.31	0.284	0.207	0.206
qi-mls	0.1	0.4	0.231	0.092	0.302	0.314	0.277	0.198	0.201
qi-mls	0.1	0.5	0.225	0.091	0.294	0.305	0.272	0.193	0.195
qi-mls	0.1	0.6	0.223	0.091	0.288	0.304	0.269	0.193	0.193
qi-mls	0.1	0.7	0.22	0.091	0.284	0.304	0.267	0.186	0.191
qi-mls	0.1	0.8	0.219	0.09	0.277	0.303	0.266	0.184	0.19
qi-mls	0.1	0.9	0.22	0.091	0.274	0.305	0.27	0.188	0.191
qi-mls	0.3	—	0.233	0.09	0.219	0.178	0.315	0.342	0.255
qi-mls	0.3	0.1	0.157	0.073	0.232	0.176	0.091	0.19	0.178
qi-mls	0.3	0.2	0.139	0.064	0.209	0.165	0.072	0.165	0.161
qi-mls	0.3	0.3	0.129	0.06	0.193	0.159	0.061	0.151	0.151
qi-mls	0.3	0.4	0.123	0.059	0.181	0.156	0.054	0.142	0.145
qi-mls	0.3	0.5	0.118	0.058	0.172	0.154	0.05	0.136	0.14
qi-mls	0.3	0.6	0.115	0.057	0.165	0.153	0.046	0.131	0.137
qi-mls	0.3	0.7	0.113	0.057	0.16	0.152	0.044	0.129	0.135
qi-mls	0.3	0.8	0.112	0.057	0.156	0.152	0.043	0.127	0.134
qi-mls	0.3	0.9	0.112	0.057	0.155	0.153	0.045	0.129	0.134
qi-mls	0.5	—	0.075	0.029	0.108	0.102	0.037	0.1	0.074
qi-mls	0.5	0.1	0.115	0.044	0.169	0.118	0.075	0.152	0.135
qi-mls	0.5	0.2	0.098	0.035	0.145	0.107	0.055	0.126	0.117
qi-mls	0.5	0.3	0.087	0.031	0.128	0.101	0.044	0.112	0.108
qi-mls	0.5	0.4	0.08	0.029	0.116	0.098	0.037	0.102	0.101
qi-mls	0.5	0.5	0.076	0.028	0.106	0.095	0.032	0.096	0.096
qi-mls	0.5	0.6	0.072	0.028	0.099	0.094	0.029	0.091	0.093
qi-mls	0.5	0.7	0.07	0.028	0.093	0.093	0.026	0.088	0.091
qi-mls	0.5	0.8	0.068	0.027	0.088	0.093	0.024	0.085	0.089
qi-mls	0.5	0.9	0.067	0.027	0.085	0.093	0.024	0.084	0.088
qi-mls	0.7	—	0.075	0.029	0.108	0.102	0.037	0.1	0.074
qi-mls	0.7	0.1	0.115	0.044	0.169	0.118	0.075	0.152	0.135
qi-mls	0.7	0.2	0.098	0.035	0.145	0.107	0.055	0.126	0.117
qi-mls	0.7	0.3	0.087	0.031	0.128	0.101	0.044	0.112	0.108
qi-mls	0.7	0.4	0.08	0.029	0.116	0.098	0.037	0.102	0.101
qi-mls	0.7	0.5	0.076	0.028	0.106	0.095	0.032	0.096	0.096
qi-mls	0.7	0.6	0.072	0.028	0.099	0.094	0.029	0.091	0.093
qi-mls	0.7	0.7	0.07	0.028	0.093	0.093	0.026	0.088	0.091
qi-mls	0.7	0.8	0.068	0.027	0.088	0.093	0.024	0.085	0.089
qi-mls	0.7	0.9	0.067	0.027	0.085	0.093	0.024	0.084	0.088
qi-mls	0.9	—	0.075	0.029	0.108	0.102	0.037	0.1	0.074
qi-mls	0.9	0.1	0.115	0.044	0.169	0.118	0.075	0.152	0.135
qi-mls	0.9	0.2	0.098	0.035	0.145	0.107	0.055	0.126	0.117
qi-mls	0.9	0.3	0.087	0.031	0.128	0.101	0.044	0.112	0.108
qi-mls	0.9	0.4	0.08	0.029	0.116	0.098	0.037	0.102	0.101
qi-mls	0.9	0.5	0.076	0.028	0.106	0.095	0.032	0.096	0.096
qi-mls	0.9	0.6	0.072	0.028	0.099	0.094	0.029	0.091	0.093
qi-mls	0.9	0.7	0.07	0.028	0.093	0.093	0.026	0.088	0.091
qi-mls	0.9	0.8	0.068	0.027	0.088	0.093	0.024	0.085	0.089
qi-mls	0.9	0.9	0.067	0.027	0.085	0.093	0.024	0.084	0.088
qi-mls-ng	0.1	—	6.023	7.394	6.528	6.361	4.598	5.554	5.703
qi-mls-ng	0.1	0.1	6.052	7.387	6.528	6.365	4.638	5.551	5.842
qi-mls-ng	0.1	0.2	6.052	7.386	6.528	6.365	4.637	5.551	5.842
qi-mls-ng	0.1	0.3	6.035	7.386	6.528	6.365	4.637	5.551	5.745
qi-mls-ng	0.1	0.4	6.031	7.386	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.1	0.5	6.024	7.345	6.528	6.365	4.61	5.551	5.745
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 20 (VOI_m on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-ng	0.1	0.6	6.023	7.345	6.528	6.365	4.603	5.552	5.745
qi-mls-ng	0.1	0.7	6.018	7.345	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.1	0.8	6.018	7.345	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.1	0.9	6.026	7.392	6.528	6.365	4.603	5.553	5.718
qi-mls-ng	0.3	—	6.014	7.343	6.528	6.361	4.598	5.554	5.703
qi-mls-ng	0.3	0.1	6.044	7.341	6.528	6.365	4.638	5.551	5.842
qi-mls-ng	0.3	0.2	6.044	7.34	6.528	6.365	4.637	5.551	5.842
qi-mls-ng	0.3	0.3	6.028	7.34	6.528	6.365	4.637	5.551	5.745
qi-mls-ng	0.3	0.4	6.023	7.34	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.3	0.5	6.02	7.323	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.3	0.6	6.019	7.323	6.528	6.365	4.603	5.552	5.745
qi-mls-ng	0.3	0.7	6.015	7.323	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.3	0.8	6.015	7.323	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.3	0.9	6.015	7.322	6.528	6.365	4.603	5.553	5.718
qi-mls-ng	0.5	—	6.014	7.343	6.528	6.361	4.598	5.554	5.703
qi-mls-ng	0.5	0.1	6.044	7.341	6.528	6.365	4.638	5.551	5.842
qi-mls-ng	0.5	0.2	6.044	7.34	6.528	6.365	4.637	5.551	5.842
qi-mls-ng	0.5	0.3	6.028	7.34	6.528	6.365	4.637	5.551	5.745
qi-mls-ng	0.5	0.4	6.023	7.34	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.5	0.5	6.02	7.323	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.5	0.6	6.019	7.323	6.528	6.365	4.603	5.552	5.745
qi-mls-ng	0.5	0.7	6.015	7.323	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.5	0.8	6.015	7.323	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.5	0.9	6.015	7.322	6.528	6.365	4.603	5.553	5.718
qi-mls-ng	0.7	—	6.014	7.343	6.528	6.361	4.598	5.554	5.703
qi-mls-ng	0.7	0.1	6.044	7.341	6.528	6.365	4.638	5.551	5.842
qi-mls-ng	0.7	0.2	6.044	7.34	6.528	6.365	4.637	5.551	5.842
qi-mls-ng	0.7	0.3	6.028	7.34	6.528	6.365	4.637	5.551	5.745
qi-mls-ng	0.7	0.4	6.023	7.34	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.7	0.5	6.02	7.323	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.7	0.6	6.019	7.323	6.528	6.365	4.603	5.552	5.745
qi-mls-ng	0.7	0.7	6.015	7.323	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.7	0.8	6.015	7.323	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.7	0.9	6.015	7.322	6.528	6.365	4.603	5.553	5.718
qi-mls-ng	0.9	—	6.014	7.343	6.528	6.361	4.598	5.554	5.703
qi-mls-ng	0.9	0.1	6.044	7.341	6.528	6.365	4.638	5.551	5.842
qi-mls-ng	0.9	0.2	6.044	7.34	6.528	6.365	4.637	5.551	5.842
qi-mls-ng	0.9	0.3	6.028	7.34	6.528	6.365	4.637	5.551	5.745
qi-mls-ng	0.9	0.4	6.023	7.34	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.9	0.5	6.02	7.323	6.528	6.365	4.61	5.551	5.745
qi-mls-ng	0.9	0.6	6.019	7.323	6.528	6.365	4.603	5.552	5.745
qi-mls-ng	0.9	0.7	6.015	7.323	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.9	0.8	6.015	7.323	6.528	6.365	4.603	5.552	5.718
qi-mls-ng	0.9	0.9	6.015	7.322	6.528	6.365	4.603	5.553	5.718
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Table 21: RAND performance on 100% of the data including training data for all architectures and parameter sets. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging.

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	—	0.92	0.988	0.961	0.921	0.813	0.901	0.933
qi-mse	0.1	0.1	0.926	0.989	0.961	0.921	0.79	0.953	0.944
qi-mse	0.1	0.2	0.928	0.989	0.962	0.921	0.79	0.953	0.951
qi-mse	0.1	0.3	0.928	0.989	0.962	0.921	0.79	0.955	0.951
qi-mse	0.1	0.4	0.928	0.989	0.962	0.921	0.79	0.955	0.952
qi-mse	0.1	0.5	0.928	0.989	0.962	0.921	0.79	0.955	0.952
qi-mse	0.1	0.6	0.928	0.989	0.962	0.921	0.79	0.955	0.952
qi-mse	0.1	0.7	0.928	0.989	0.962	0.921	0.79	0.955	0.952
qi-mse	0.1	0.8	0.928	0.989	0.962	0.921	0.79	0.955	0.952
qi-mse	0.1	0.9	0.928	0.989	0.962	0.921	0.791	0.955	0.952
qi-mse	0.3	—	0.9	0.989	0.961	0.923	0.677	0.917	0.931
qi-mse	0.3	0.1	0.9	0.989	0.961	0.92	0.634	0.953	0.944
qi-mse	0.3	0.2	0.9	0.989	0.961	0.921	0.634	0.953	0.944
qi-mse	0.3	0.3	0.9	0.989	0.961	0.92	0.634	0.953	0.944
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 21 (RAND on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.3	0.4	0.9	0.989	0.961	0.92	0.634	0.953	0.944
qi-mse	0.3	0.5	0.9	0.989	0.961	0.92	0.634	0.953	0.944
qi-mse	0.3	0.6	0.9	0.989	0.961	0.92	0.633	0.953	0.944
qi-mse	0.3	0.7	0.9	0.989	0.962	0.92	0.633	0.953	0.944
qi-mse	0.3	0.8	0.9	0.989	0.962	0.92	0.633	0.953	0.944
qi-mse	0.3	0.9	0.9	0.989	0.962	0.92	0.633	0.953	0.944
qi-mse	0.5	—	0.642	0.613	0.88	0.515	0.315	0.846	0.684
qi-mse	0.5	0.1	0.664	0.657	0.882	0.626	0.249	0.827	0.741
qi-mse	0.5	0.2	0.657	0.627	0.88	0.621	0.248	0.827	0.737
qi-mse	0.5	0.3	0.663	0.666	0.879	0.617	0.247	0.834	0.733
qi-mse	0.5	0.4	0.658	0.63	0.878	0.617	0.247	0.833	0.74
qi-mse	0.5	0.5	0.658	0.647	0.878	0.619	0.246	0.827	0.729
qi-mse	0.5	0.6	0.65	0.625	0.86	0.616	0.245	0.827	0.729
qi-mse	0.5	0.7	0.651	0.629	0.86	0.618	0.245	0.827	0.729
qi-mse	0.5	0.8	0.652	0.63	0.86	0.615	0.245	0.833	0.729
qi-mse	0.5	0.9	0.634	0.628	0.859	0.615	0.243	0.824	0.637
qi-mse	0.7	—	0.369	0.187	0.153	0.4	0.279	0.731	0.464
qi-mse	0.7	0.1	0.357	0.168	0.163	0.398	0.226	0.732	0.457
qi-mse	0.7	0.2	0.356	0.167	0.163	0.398	0.223	0.731	0.456
qi-mse	0.7	0.3	0.356	0.167	0.163	0.398	0.222	0.732	0.456
qi-mse	0.7	0.4	0.356	0.167	0.16	0.398	0.222	0.732	0.456
qi-mse	0.7	0.5	0.356	0.167	0.16	0.398	0.222	0.732	0.456
qi-mse	0.7	0.6	0.356	0.167	0.16	0.398	0.222	0.732	0.456
qi-mse	0.7	0.7	0.356	0.167	0.16	0.398	0.222	0.732	0.456
qi-mse	0.7	0.8	0.356	0.167	0.16	0.398	0.222	0.732	0.456
qi-mse	0.7	0.9	0.356	0.168	0.16	0.398	0.222	0.732	0.457
qi-mse	0.9	—	0.475	0.618	0.21	0.455	0.289	0.759	0.517
qi-mse	0.9	0.1	0.464	0.6	0.213	0.454	0.236	0.767	0.516
qi-mse	0.9	0.2	0.464	0.6	0.212	0.454	0.235	0.767	0.518
qi-mse	0.9	0.3	0.464	0.6	0.212	0.454	0.235	0.767	0.518
qi-mse	0.9	0.4	0.465	0.604	0.211	0.454	0.234	0.767	0.518
qi-mse	0.9	0.5	0.465	0.605	0.211	0.454	0.234	0.767	0.519
qi-mse	0.9	0.6	0.466	0.606	0.211	0.454	0.234	0.769	0.519
qi-mse	0.9	0.7	0.466	0.608	0.211	0.454	0.235	0.769	0.52
qi-mse	0.9	0.8	0.466	0.608	0.211	0.454	0.235	0.77	0.52
qi-mse	0.9	0.9	0.467	0.608	0.211	0.454	0.235	0.77	0.52
qi-mse-ng	0.1	—	0.916	0.988	0.961	0.921	0.817	0.875	0.934
qi-mse-ng	0.1	0.1	0.926	0.989	0.961	0.92	0.779	0.955	0.951
qi-mse-ng	0.1	0.2	0.928	0.989	0.962	0.92	0.79	0.955	0.951
qi-mse-ng	0.1	0.3	0.928	0.989	0.962	0.92	0.79	0.955	0.951
qi-mse-ng	0.1	0.4	0.928	0.989	0.962	0.921	0.79	0.955	0.951
qi-mse-ng	0.1	0.5	0.928	0.989	0.962	0.921	0.791	0.955	0.951
qi-mse-ng	0.1	0.6	0.928	0.989	0.962	0.921	0.791	0.955	0.951
qi-mse-ng	0.1	0.7	0.928	0.989	0.962	0.921	0.791	0.955	0.951
qi-mse-ng	0.1	0.8	0.928	0.989	0.962	0.921	0.791	0.955	0.951
qi-mse-ng	0.1	0.9	0.928	0.989	0.962	0.921	0.791	0.955	0.951
qi-mse-ng	0.3	—	0.92	0.988	0.962	0.923	0.802	0.907	0.939
qi-mse-ng	0.3	0.1	0.926	0.989	0.961	0.92	0.779	0.955	0.951
qi-mse-ng	0.3	0.2	0.928	0.989	0.962	0.92	0.79	0.955	0.951
qi-mse-ng	0.3	0.3	0.928	0.989	0.962	0.92	0.79	0.955	0.951
qi-mse-ng	0.3	0.4	0.928	0.989	0.962	0.92	0.79	0.955	0.951
qi-mse-ng	0.3	0.5	0.927	0.989	0.962	0.921	0.79	0.955	0.944
qi-mse-ng	0.3	0.6	0.927	0.989	0.962	0.921	0.79	0.955	0.944
qi-mse-ng	0.3	0.7	0.927	0.989	0.962	0.921	0.79	0.955	0.944
qi-mse-ng	0.3	0.8	0.926	0.989	0.962	0.914	0.79	0.955	0.944
qi-mse-ng	0.3	0.9	0.926	0.989	0.962	0.914	0.79	0.955	0.944
qi-mse-ng	0.5	—	0.863	0.976	0.955	0.851	0.565	0.913	0.918
qi-mse-ng	0.5	0.1	0.874	0.976	0.955	0.849	0.594	0.939	0.928
qi-mse-ng	0.5	0.2	0.859	0.977	0.955	0.844	0.514	0.938	0.928
qi-mse-ng	0.5	0.3	0.861	0.976	0.955	0.855	0.514	0.938	0.927
qi-mse-ng	0.5	0.4	0.859	0.975	0.955	0.846	0.512	0.938	0.928
qi-mse-ng	0.5	0.5	0.856	0.975	0.955	0.842	0.51	0.941	0.912
qi-mse-ng	0.5	0.6	0.855	0.974	0.955	0.842	0.509	0.938	0.912
qi-mse-ng	0.5	0.7	0.854	0.974	0.955	0.836	0.506	0.938	0.912
qi-mse-ng	0.5	0.8	0.855	0.974	0.955	0.848	0.506	0.938	0.911
qi-mse-ng	0.5	0.9	0.855	0.974	0.955	0.847	0.507	0.938	0.911
qi-mse-ng	0.7	—	0.376	0.181	0.2	0.405	0.291	0.717	0.46
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 21 (RAND on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse-ng	0.7	0.1	0.364	0.165	0.203	0.404	0.225	0.733	0.456
qi-mse-ng	0.7	0.2	0.363	0.161	0.202	0.403	0.224	0.733	0.455
qi-mse-ng	0.7	0.3	0.363	0.16	0.201	0.403	0.224	0.732	0.455
qi-mse-ng	0.7	0.4	0.363	0.164	0.201	0.403	0.224	0.732	0.455
qi-mse-ng	0.7	0.5	0.363	0.165	0.201	0.403	0.224	0.732	0.455
qi-mse-ng	0.7	0.6	0.363	0.165	0.201	0.403	0.223	0.732	0.455
qi-mse-ng	0.7	0.7	0.363	0.165	0.201	0.403	0.223	0.733	0.455
qi-mse-ng	0.7	0.8	0.36	0.165	0.18	0.403	0.224	0.733	0.455
qi-mse-ng	0.7	0.9	0.36	0.165	0.181	0.403	0.224	0.733	0.455
qi-mse-ng	0.9	—	0.453	0.55	0.192	0.441	0.29	0.747	0.498
qi-mse-ng	0.9	0.1	0.437	0.516	0.183	0.439	0.233	0.759	0.495
qi-mse-ng	0.9	0.2	0.437	0.518	0.182	0.439	0.232	0.759	0.495
qi-mse-ng	0.9	0.3	0.438	0.519	0.182	0.439	0.232	0.759	0.495
qi-mse-ng	0.9	0.4	0.438	0.522	0.181	0.44	0.232	0.759	0.495
qi-mse-ng	0.9	0.5	0.44	0.523	0.187	0.44	0.232	0.76	0.496
qi-mse-ng	0.9	0.6	0.44	0.524	0.188	0.44	0.232	0.76	0.496
qi-mse-ng	0.9	0.7	0.441	0.527	0.188	0.44	0.232	0.763	0.497
qi-mse-ng	0.9	0.8	0.441	0.527	0.188	0.44	0.232	0.763	0.498
qi-mse-ng	0.9	0.9	0.443	0.533	0.189	0.44	0.232	0.763	0.498
qi-mls-pre	0.1	—	0.847	0.951	0.924	0.803	0.75	0.754	0.899
qi-mls-pre	0.1	0.1	0.853	0.97	0.928	0.807	0.584	0.919	0.91
qi-mls-pre	0.1	0.2	0.85	0.97	0.928	0.808	0.565	0.92	0.91
qi-mls-pre	0.1	0.3	0.849	0.97	0.929	0.808	0.563	0.919	0.907
qi-mls-pre	0.1	0.4	0.851	0.97	0.929	0.807	0.57	0.921	0.907
qi-mls-pre	0.1	0.5	0.853	0.97	0.928	0.807	0.595	0.907	0.907
qi-mls-pre	0.1	0.6	0.836	0.971	0.928	0.807	0.496	0.908	0.907
qi-mls-pre	0.1	0.7	0.837	0.971	0.928	0.807	0.498	0.909	0.907
qi-mls-pre	0.1	0.8	0.845	0.971	0.928	0.807	0.549	0.908	0.907
qi-mls-pre	0.1	0.9	0.845	0.971	0.927	0.807	0.551	0.907	0.907
qi-mls-pre	0.3	—	0.957	0.964	0.96	0.979	0.943	0.948	0.951
qi-mls-pre	0.3	0.1	0.978	0.966	0.964	0.983	0.976	0.99	0.988
qi-mls-pre	0.3	0.2	0.978	0.966	0.963	0.983	0.976	0.99	0.988
qi-mls-pre	0.3	0.3	0.978	0.966	0.963	0.983	0.976	0.99	0.988
qi-mls-pre	0.3	0.4	0.978	0.966	0.963	0.983	0.976	0.989	0.988
qi-mls-pre	0.3	0.5	0.978	0.966	0.963	0.983	0.976	0.989	0.988
qi-mls-pre	0.3	0.6	0.974	0.966	0.963	0.983	0.955	0.989	0.988
qi-mls-pre	0.3	0.7	0.974	0.966	0.963	0.983	0.955	0.989	0.988
qi-mls-pre	0.3	0.8	0.974	0.966	0.963	0.983	0.955	0.989	0.988
qi-mls-pre	0.3	0.9	0.974	0.966	0.962	0.983	0.955	0.989	0.988
qi-mls-pre	0.5	—	0.959	0.965	0.965	0.981	0.944	0.951	0.952
qi-mls-pre	0.5	0.1	0.979	0.967	0.966	0.984	0.977	0.991	0.989
qi-mls-pre	0.5	0.2	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.5	0.3	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.5	0.4	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.5	0.5	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.5	0.6	0.975	0.967	0.965	0.984	0.955	0.991	0.988
qi-mls-pre	0.5	0.7	0.975	0.967	0.965	0.984	0.955	0.991	0.988
qi-mls-pre	0.5	0.8	0.975	0.967	0.965	0.984	0.955	0.991	0.988
qi-mls-pre	0.5	0.9	0.975	0.967	0.964	0.984	0.955	0.991	0.988
qi-mls-pre	0.7	—	0.959	0.965	0.965	0.981	0.944	0.951	0.952
qi-mls-pre	0.7	0.1	0.979	0.967	0.966	0.984	0.977	0.991	0.989
qi-mls-pre	0.7	0.2	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.7	0.3	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.7	0.4	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.7	0.5	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.7	0.6	0.975	0.967	0.965	0.984	0.955	0.991	0.988
qi-mls-pre	0.7	0.7	0.975	0.967	0.965	0.984	0.955	0.991	0.988
qi-mls-pre	0.7	0.8	0.975	0.967	0.965	0.984	0.955	0.991	0.988
qi-mls-pre	0.7	0.9	0.975	0.967	0.964	0.984	0.955	0.991	0.988
qi-mls-pre	0.9	—	0.959	0.965	0.965	0.981	0.944	0.951	0.952
qi-mls-pre	0.9	0.1	0.979	0.967	0.966	0.984	0.977	0.991	0.989
qi-mls-pre	0.9	0.2	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.9	0.3	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.9	0.4	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.9	0.5	0.979	0.967	0.965	0.984	0.976	0.991	0.989
qi-mls-pre	0.9	0.6	0.975	0.967	0.965	0.984	0.955	0.991	0.988
qi-mls-pre	0.9	0.7	0.975	0.967	0.965	0.984	0.955	0.991	0.988
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 21 (RAND on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre	0.9	0.8	0.975	0.967	0.965	0.984	0.955	0.991	0.988
qi-mls-pre	0.9	0.9	0.975	0.967	0.964	0.984	0.955	0.991	0.988
qi-mls-pre-ng	0.1	—	0.908	0.986	0.954	0.903	0.845	0.848	0.914
qi-mls-pre-ng	0.1	0.1	0.943	0.987	0.952	0.908	0.949	0.919	0.943
qi-mls-pre-ng	0.1	0.2	0.947	0.987	0.952	0.906	0.949	0.944	0.943
qi-mls-pre-ng	0.1	0.3	0.946	0.987	0.951	0.906	0.949	0.938	0.943
qi-mls-pre-ng	0.1	0.4	0.945	0.987	0.951	0.905	0.948	0.938	0.943
qi-mls-pre-ng	0.1	0.5	0.945	0.987	0.951	0.905	0.948	0.938	0.943
qi-mls-pre-ng	0.1	0.6	0.945	0.987	0.951	0.905	0.948	0.938	0.943
qi-mls-pre-ng	0.1	0.7	0.943	0.987	0.951	0.905	0.935	0.938	0.943
qi-mls-pre-ng	0.1	0.8	0.943	0.987	0.951	0.904	0.935	0.937	0.943
qi-mls-pre-ng	0.1	0.9	0.938	0.987	0.952	0.905	0.904	0.936	0.944
qi-mls-pre-ng	0.3	—	0.845	0.817	0.784	0.841	0.917	0.871	0.841
qi-mls-pre-ng	0.3	0.1	0.88	0.807	0.793	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.3	0.2	0.879	0.807	0.792	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.3	0.3	0.879	0.807	0.792	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.3	0.4	0.879	0.807	0.792	0.852	0.966	0.95	0.907
qi-mls-pre-ng	0.3	0.5	0.879	0.807	0.792	0.852	0.966	0.949	0.907
qi-mls-pre-ng	0.3	0.6	0.879	0.808	0.792	0.852	0.966	0.949	0.907
qi-mls-pre-ng	0.3	0.7	0.878	0.808	0.792	0.852	0.958	0.949	0.907
qi-mls-pre-ng	0.3	0.8	0.878	0.808	0.792	0.853	0.958	0.949	0.906
qi-mls-pre-ng	0.3	0.9	0.871	0.808	0.792	0.853	0.939	0.948	0.888
qi-mls-pre-ng	0.5	—	0.845	0.817	0.784	0.841	0.917	0.871	0.841
qi-mls-pre-ng	0.5	0.1	0.88	0.807	0.793	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.5	0.2	0.879	0.807	0.792	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.5	0.3	0.879	0.807	0.792	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.5	0.4	0.879	0.807	0.792	0.852	0.966	0.95	0.907
qi-mls-pre-ng	0.5	0.5	0.879	0.807	0.792	0.852	0.966	0.949	0.907
qi-mls-pre-ng	0.5	0.6	0.879	0.808	0.792	0.852	0.966	0.949	0.907
qi-mls-pre-ng	0.5	0.7	0.878	0.808	0.792	0.852	0.958	0.949	0.907
qi-mls-pre-ng	0.5	0.8	0.878	0.808	0.792	0.853	0.958	0.949	0.906
qi-mls-pre-ng	0.5	0.9	0.871	0.808	0.792	0.853	0.939	0.948	0.888
qi-mls-pre-ng	0.7	—	0.845	0.817	0.784	0.841	0.917	0.871	0.841
qi-mls-pre-ng	0.7	0.1	0.88	0.807	0.793	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.7	0.2	0.879	0.807	0.792	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.7	0.3	0.879	0.807	0.792	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.7	0.4	0.879	0.807	0.792	0.852	0.966	0.95	0.907
qi-mls-pre-ng	0.7	0.5	0.879	0.807	0.792	0.852	0.966	0.949	0.907
qi-mls-pre-ng	0.7	0.6	0.879	0.808	0.792	0.852	0.966	0.949	0.907
qi-mls-pre-ng	0.7	0.7	0.878	0.808	0.792	0.852	0.958	0.949	0.907
qi-mls-pre-ng	0.7	0.8	0.878	0.808	0.792	0.853	0.958	0.949	0.906
qi-mls-pre-ng	0.7	0.9	0.871	0.808	0.792	0.853	0.939	0.948	0.888
qi-mls-pre-ng	0.9	—	0.845	0.817	0.784	0.841	0.917	0.871	0.841
qi-mls-pre-ng	0.9	0.1	0.88	0.807	0.793	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.9	0.2	0.879	0.807	0.792	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.9	0.3	0.879	0.807	0.792	0.852	0.967	0.952	0.907
qi-mls-pre-ng	0.9	0.4	0.879	0.807	0.792	0.852	0.966	0.95	0.907
qi-mls-pre-ng	0.9	0.5	0.879	0.807	0.792	0.852	0.966	0.949	0.907
qi-mls-pre-ng	0.9	0.6	0.879	0.808	0.792	0.852	0.966	0.949	0.907
qi-mls-pre-ng	0.9	0.7	0.878	0.808	0.792	0.852	0.958	0.949	0.907
qi-mls-pre-ng	0.9	0.8	0.878	0.808	0.792	0.853	0.958	0.949	0.906
qi-mls-pre-ng	0.9	0.9	0.871	0.808	0.792	0.853	0.939	0.948	0.888
qi-mls	0.1	—	0.387	0.25	0.141	0.422	0.41	0.642	0.457
qi-mls	0.1	0.1	0.39	0.24	0.135	0.429	0.33	0.738	0.467
qi-mls	0.1	0.2	0.389	0.24	0.134	0.429	0.328	0.737	0.467
qi-mls	0.1	0.3	0.389	0.241	0.136	0.429	0.328	0.736	0.466
qi-mls	0.1	0.4	0.388	0.242	0.132	0.429	0.327	0.736	0.466
qi-mls	0.1	0.5	0.389	0.242	0.131	0.428	0.327	0.738	0.466
qi-mls	0.1	0.6	0.389	0.242	0.131	0.428	0.326	0.738	0.466
qi-mls	0.1	0.7	0.389	0.242	0.131	0.428	0.326	0.738	0.465
qi-mls	0.1	0.8	0.389	0.243	0.131	0.428	0.326	0.743	0.465
qi-mls	0.1	0.9	0.39	0.244	0.131	0.429	0.326	0.743	0.466
qi-mls	0.3	—	0.575	0.628	0.394	0.65	0.368	0.78	0.631
qi-mls	0.3	0.1	0.587	0.628	0.395	0.662	0.345	0.816	0.673
qi-mls	0.3	0.2	0.587	0.629	0.395	0.662	0.346	0.817	0.673
qi-mls	0.3	0.3	0.587	0.629	0.393	0.662	0.346	0.818	0.673
qi-mls	0.3	0.4	0.587	0.63	0.393	0.662	0.347	0.818	0.673
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 21 (RAND on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls	0.3	0.5	0.587	0.63	0.393	0.662	0.347	0.818	0.674
qi-mls	0.3	0.6	0.588	0.632	0.393	0.662	0.35	0.818	0.674
qi-mls	0.3	0.7	0.589	0.633	0.394	0.662	0.351	0.818	0.675
qi-mls	0.3	0.8	0.589	0.635	0.394	0.662	0.352	0.819	0.675
qi-mls	0.3	0.9	0.59	0.638	0.394	0.662	0.352	0.819	0.676
qi-mls	0.5	—	0.959	0.989	0.976	0.982	0.966	0.911	0.932
qi-mls	0.5	0.1	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.5	0.2	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.5	0.3	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.5	0.4	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.5	0.5	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.5	0.6	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.5	0.7	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.5	0.8	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.5	0.9	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	—	0.959	0.989	0.976	0.982	0.966	0.911	0.932
qi-mls	0.7	0.1	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	0.2	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	0.3	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	0.4	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	0.5	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	0.6	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	0.7	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	0.8	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.7	0.9	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	—	0.959	0.989	0.976	0.982	0.966	0.911	0.932
qi-mls	0.9	0.1	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	0.2	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	0.3	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	0.4	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	0.5	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	0.6	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	0.7	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	0.8	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls	0.9	0.9	1.0	0.999	1.0	1.0	1.0	1.0	1.0
qi-mls-ng	0.1	—	0.989	0.97	0.992	0.997	0.998	0.998	0.979
qi-mls-ng	0.1	0.1	0.992	0.969	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.1	0.2	0.992	0.969	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.1	0.3	0.992	0.969	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.1	0.4	0.992	0.969	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.1	0.5	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.1	0.6	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.1	0.7	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.1	0.8	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.1	0.9	0.992	0.97	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.3	—	0.988	0.966	0.992	0.997	0.998	0.998	0.979
qi-mls-ng	0.3	0.1	0.992	0.966	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.3	0.2	0.992	0.966	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.3	0.3	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.3	0.4	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.3	0.5	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.3	0.6	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.3	0.7	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.3	0.8	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.3	0.9	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.5	—	0.988	0.966	0.992	0.997	0.998	0.998	0.979
qi-mls-ng	0.5	0.1	0.992	0.966	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.5	0.2	0.992	0.966	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.5	0.3	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.5	0.4	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.5	0.5	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.5	0.6	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.5	0.7	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.5	0.8	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.5	0.9	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.7	—	0.988	0.966	0.992	0.997	0.998	0.998	0.979
qi-mls-ng	0.7	0.1	0.992	0.966	0.992	0.997	0.999	1.0	0.996
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 21 (RAND on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-ng	0.7	0.2	0.992	0.966	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.7	0.3	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.7	0.4	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.7	0.5	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.7	0.6	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.7	0.7	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.7	0.8	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.7	0.9	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.9	—	0.988	0.966	0.992	0.997	0.998	0.998	0.979
qi-mls-ng	0.9	0.1	0.992	0.966	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.9	0.2	0.992	0.966	0.992	0.997	0.999	1.0	0.996
qi-mls-ng	0.9	0.3	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.9	0.4	0.991	0.966	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.9	0.5	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.9	0.6	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.9	0.7	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.9	0.8	0.991	0.965	0.992	0.997	0.999	1.0	0.994
qi-mls-ng	0.9	0.9	0.991	0.965	0.992	0.997	0.999	1.0	0.994
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Table 22: CREMI score performance on 100% of the data including training data for all architectures and parameter sets. A "—" in the t_g column means that the glia predictions were not considered during super voxel generation and merging.

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.1	—	2.425	2.762	2.519	2.499	2.01	2.35	2.409
qi-mse	0.1	0.1	2.278	2.737	2.485	2.429	1.727	2.121	2.17
qi-mse	0.1	0.2	2.288	2.735	2.481	2.427	1.726	2.118	2.24
qi-mse	0.1	0.3	2.29	2.734	2.479	2.427	1.724	2.138	2.239
qi-mse	0.1	0.4	2.292	2.734	2.478	2.426	1.723	2.138	2.251
qi-mse	0.1	0.5	2.292	2.734	2.478	2.427	1.723	2.138	2.251
qi-mse	0.1	0.6	2.292	2.734	2.478	2.427	1.723	2.138	2.251
qi-mse	0.1	0.7	2.292	2.734	2.478	2.427	1.722	2.139	2.251
qi-mse	0.1	0.8	2.292	2.734	2.478	2.428	1.719	2.14	2.251
qi-mse	0.1	0.9	2.293	2.735	2.48	2.43	1.721	2.143	2.252
qi-mse	0.3	—	2.372	2.765	2.523	2.514	1.67	2.405	2.357
qi-mse	0.3	0.1	2.212	2.737	2.482	2.419	1.347	2.117	2.168
qi-mse	0.3	0.2	2.21	2.735	2.478	2.425	1.342	2.113	2.165
qi-mse	0.3	0.3	2.206	2.727	2.476	2.416	1.34	2.111	2.163
qi-mse	0.3	0.4	2.205	2.726	2.475	2.416	1.339	2.111	2.163
qi-mse	0.3	0.5	2.205	2.726	2.474	2.416	1.338	2.111	2.162
qi-mse	0.3	0.6	2.204	2.726	2.473	2.416	1.333	2.111	2.162
qi-mse	0.3	0.7	2.204	2.727	2.473	2.416	1.333	2.112	2.162
qi-mse	0.3	0.8	2.204	2.727	2.474	2.417	1.334	2.113	2.162
qi-mse	0.3	0.9	2.206	2.727	2.476	2.419	1.335	2.115	2.163
qi-mse	0.5	—	1.366	1.055	1.77	1.2	0.832	1.906	1.433
qi-mse	0.5	0.1	1.156	1.079	1.724	1.247	0.437	1.245	1.201
qi-mse	0.5	0.2	1.131	1.016	1.693	1.231	0.426	1.237	1.18
qi-mse	0.5	0.3	1.136	1.062	1.686	1.218	0.419	1.272	1.157
qi-mse	0.5	0.4	1.132	1.019	1.681	1.215	0.42	1.269	1.19
qi-mse	0.5	0.5	1.121	1.04	1.678	1.22	0.413	1.231	1.146
qi-mse	0.5	0.6	1.104	1.005	1.608	1.216	0.411	1.241	1.146
qi-mse	0.5	0.7	1.106	1.009	1.608	1.22	0.411	1.242	1.145
qi-mse	0.5	0.8	1.11	1.012	1.609	1.215	0.412	1.263	1.15
qi-mse	0.5	0.9	1.081	1.012	1.612	1.218	0.404	1.231	1.006
qi-mse	0.7	—	0.851	0.411	0.471	1.009	0.735	1.486	0.991
qi-mse	0.7	0.1	0.532	0.318	0.438	0.875	0.302	0.697	0.562
qi-mse	0.7	0.2	0.524	0.313	0.433	0.873	0.284	0.685	0.554
qi-mse	0.7	0.3	0.523	0.312	0.429	0.872	0.281	0.69	0.552
qi-mse	0.7	0.4	0.522	0.312	0.425	0.873	0.28	0.692	0.551
qi-mse	0.7	0.5	0.522	0.312	0.423	0.873	0.28	0.693	0.551
qi-mse	0.7	0.6	0.522	0.312	0.423	0.874	0.28	0.694	0.551
qi-mse	0.7	0.7	0.524	0.313	0.424	0.875	0.283	0.698	0.552
qi-mse	0.7	0.8	0.527	0.313	0.425	0.877	0.287	0.706	0.554
qi-mse	0.7	0.9	0.532	0.316	0.428	0.88	0.291	0.721	0.558
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 22 (CREMI score on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mse	0.9	—	1.276	1.609	0.748	1.325	0.907	1.748	1.319
qi-mse	0.9	0.1	1.018	1.525	0.702	1.213	0.542	1.134	0.994
qi-mse	0.9	0.2	1.016	1.525	0.697	1.211	0.538	1.13	0.996
qi-mse	0.9	0.3	1.015	1.525	0.695	1.211	0.538	1.129	0.995
qi-mse	0.9	0.4	1.016	1.532	0.694	1.211	0.537	1.129	0.995
qi-mse	0.9	0.5	1.017	1.534	0.693	1.211	0.538	1.13	0.998
qi-mse	0.9	0.6	1.019	1.536	0.693	1.212	0.539	1.135	0.999
qi-mse	0.9	0.7	1.021	1.54	0.694	1.213	0.539	1.139	1.001
qi-mse	0.9	0.8	1.024	1.541	0.695	1.214	0.541	1.146	1.005
qi-mse	0.9	0.9	1.027	1.543	0.698	1.217	0.544	1.153	1.007
qi-mse-ng	0.1	—	2.407	2.762	2.519	2.498	2.013	2.254	2.397
qi-mse-ng	0.1	0.1	2.278	2.729	2.465	2.427	1.66	2.143	2.242
qi-mse-ng	0.1	0.2	2.286	2.727	2.479	2.426	1.703	2.141	2.239
qi-mse-ng	0.1	0.3	2.287	2.734	2.478	2.426	1.706	2.14	2.238
qi-mse-ng	0.1	0.4	2.286	2.733	2.477	2.426	1.705	2.14	2.237
qi-mse-ng	0.1	0.5	2.288	2.733	2.477	2.431	1.71	2.14	2.237
qi-mse-ng	0.1	0.6	2.288	2.733	2.478	2.431	1.71	2.14	2.237
qi-mse-ng	0.1	0.7	2.289	2.733	2.478	2.432	1.71	2.141	2.237
qi-mse-ng	0.1	0.8	2.289	2.733	2.48	2.433	1.711	2.142	2.237
qi-mse-ng	0.1	0.9	2.29	2.733	2.483	2.43	1.712	2.146	2.238
qi-mse-ng	0.3	—	2.423	2.757	2.53	2.515	1.958	2.366	2.414
qi-mse-ng	0.3	0.1	2.278	2.729	2.465	2.427	1.661	2.143	2.241
qi-mse-ng	0.3	0.2	2.285	2.727	2.478	2.426	1.703	2.14	2.239
qi-mse-ng	0.3	0.3	2.285	2.726	2.477	2.425	1.702	2.139	2.238
qi-mse-ng	0.3	0.4	2.284	2.726	2.477	2.425	1.701	2.139	2.237
qi-mse-ng	0.3	0.5	2.272	2.726	2.477	2.425	1.701	2.139	2.164
qi-mse-ng	0.3	0.6	2.272	2.726	2.477	2.426	1.7	2.139	2.163
qi-mse-ng	0.3	0.7	2.272	2.726	2.477	2.426	1.7	2.14	2.163
qi-mse-ng	0.3	0.8	2.266	2.726	2.479	2.385	1.701	2.142	2.164
qi-mse-ng	0.3	0.9	2.268	2.726	2.482	2.387	1.703	2.145	2.165
qi-mse-ng	0.5	—	2.092	2.317	2.404	1.989	1.41	2.252	2.179
qi-mse-ng	0.5	0.1	1.928	2.276	2.354	1.879	1.198	1.891	1.969
qi-mse-ng	0.5	0.2	1.891	2.275	2.347	1.854	1.042	1.87	1.959
qi-mse-ng	0.5	0.3	1.891	2.247	2.345	1.893	1.04	1.871	1.952
qi-mse-ng	0.5	0.4	1.884	2.243	2.345	1.861	1.034	1.863	1.957
qi-mse-ng	0.5	0.5	1.869	2.238	2.344	1.844	1.03	1.899	1.862
qi-mse-ng	0.5	0.6	1.861	2.231	2.344	1.845	1.024	1.868	1.857
qi-mse-ng	0.5	0.7	1.856	2.225	2.339	1.821	1.018	1.869	1.861
qi-mse-ng	0.5	0.8	1.861	2.221	2.341	1.862	1.018	1.871	1.853
qi-mse-ng	0.5	0.9	1.863	2.219	2.345	1.865	1.021	1.876	1.853
qi-mse-ng	0.7	—	0.844	0.387	0.554	0.998	0.739	1.43	0.956
qi-mse-ng	0.7	0.1	0.527	0.298	0.493	0.875	0.252	0.7	0.547
qi-mse-ng	0.7	0.2	0.522	0.289	0.487	0.871	0.249	0.695	0.54
qi-mse-ng	0.7	0.3	0.519	0.288	0.485	0.871	0.243	0.689	0.539
qi-mse-ng	0.7	0.4	0.52	0.293	0.483	0.871	0.247	0.688	0.537
qi-mse-ng	0.7	0.5	0.52	0.294	0.483	0.872	0.248	0.69	0.537
qi-mse-ng	0.7	0.6	0.521	0.294	0.483	0.872	0.248	0.692	0.537
qi-mse-ng	0.7	0.7	0.523	0.294	0.483	0.873	0.249	0.702	0.538
qi-mse-ng	0.7	0.8	0.522	0.295	0.455	0.875	0.254	0.713	0.542
qi-mse-ng	0.7	0.9	0.529	0.297	0.459	0.879	0.262	0.729	0.546
qi-mse-ng	0.9	—	1.173	1.38	0.673	1.248	0.869	1.664	1.202
qi-mse-ng	0.9	0.1	0.896	1.254	0.592	1.138	0.472	1.049	0.872
qi-mse-ng	0.9	0.2	0.895	1.258	0.589	1.136	0.47	1.046	0.87
qi-mse-ng	0.9	0.3	0.895	1.259	0.588	1.136	0.47	1.048	0.869
qi-mse-ng	0.9	0.4	0.897	1.266	0.587	1.137	0.47	1.05	0.87
qi-mse-ng	0.9	0.5	0.9	1.269	0.6	1.137	0.47	1.053	0.871
qi-mse-ng	0.9	0.6	0.901	1.271	0.602	1.138	0.471	1.055	0.871
qi-mse-ng	0.9	0.7	0.906	1.278	0.603	1.14	0.473	1.066	0.877
qi-mse-ng	0.9	0.8	0.909	1.281	0.605	1.142	0.475	1.07	0.881
qi-mse-ng	0.9	0.9	0.916	1.292	0.612	1.145	0.48	1.082	0.886
qi-mls-pre	0.1	—	1.981	2.087	2.137	1.977	1.739	1.808	2.135
qi-mls-pre	0.1	0.1	1.86	2.179	2.136	1.908	1.178	1.839	1.917
qi-mls-pre	0.1	0.2	1.853	2.178	2.132	1.906	1.15	1.84	1.914
qi-mls-pre	0.1	0.3	1.847	2.177	2.129	1.906	1.147	1.832	1.893
qi-mls-pre	0.1	0.4	1.852	2.176	2.126	1.902	1.163	1.848	1.894
qi-mls-pre	0.1	0.5	1.854	2.176	2.118	1.903	1.224	1.811	1.892
qi-mls-pre	0.1	0.6	1.832	2.178	2.117	1.903	1.083	1.819	1.894
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 22 (CREMI score on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre	0.1	0.7	1.836	2.178	2.117	1.904	1.095	1.829	1.896
qi-mls-pre	0.1	0.8	1.855	2.178	2.111	1.906	1.199	1.837	1.902
qi-mls-pre	0.1	0.9	1.866	2.179	2.112	1.908	1.218	1.864	1.913
qi-mls-pre	0.3	—	2.679	2.727	2.669	2.798	2.564	2.612	2.704
qi-mls-pre	0.3	0.1	2.524	2.679	2.604	2.699	2.338	2.28	2.546
qi-mls-pre	0.3	0.2	2.522	2.676	2.599	2.698	2.338	2.279	2.545
qi-mls-pre	0.3	0.3	2.522	2.675	2.596	2.697	2.339	2.281	2.546
qi-mls-pre	0.3	0.4	2.523	2.675	2.593	2.697	2.34	2.284	2.546
qi-mls-pre	0.3	0.5	2.524	2.675	2.591	2.697	2.344	2.289	2.548
qi-mls-pre	0.3	0.6	2.518	2.675	2.59	2.698	2.297	2.296	2.55
qi-mls-pre	0.3	0.7	2.52	2.675	2.59	2.699	2.302	2.305	2.552
qi-mls-pre	0.3	0.8	2.525	2.675	2.591	2.701	2.31	2.319	2.556
qi-mls-pre	0.3	0.9	2.535	2.675	2.593	2.705	2.328	2.347	2.564
qi-mls-pre	0.5	—	2.73	2.754	2.731	2.834	2.614	2.7	2.743
qi-mls-pre	0.5	0.1	2.555	2.699	2.64	2.728	2.362	2.329	2.573
qi-mls-pre	0.5	0.2	2.553	2.697	2.635	2.726	2.362	2.328	2.571
qi-mls-pre	0.5	0.3	2.553	2.696	2.632	2.725	2.362	2.329	2.572
qi-mls-pre	0.5	0.4	2.553	2.695	2.629	2.725	2.364	2.332	2.573
qi-mls-pre	0.5	0.5	2.554	2.695	2.627	2.726	2.367	2.337	2.574
qi-mls-pre	0.5	0.6	2.548	2.695	2.626	2.727	2.321	2.344	2.576
qi-mls-pre	0.5	0.7	2.551	2.695	2.626	2.728	2.325	2.353	2.578
qi-mls-pre	0.5	0.8	2.555	2.695	2.627	2.729	2.333	2.367	2.582
qi-mls-pre	0.5	0.9	2.565	2.695	2.629	2.733	2.351	2.394	2.59
qi-mls-pre	0.7	—	2.73	2.754	2.731	2.834	2.614	2.7	2.743
qi-mls-pre	0.7	0.1	2.555	2.699	2.64	2.728	2.362	2.329	2.573
qi-mls-pre	0.7	0.2	2.553	2.697	2.635	2.726	2.362	2.328	2.571
qi-mls-pre	0.7	0.3	2.553	2.696	2.632	2.725	2.362	2.329	2.572
qi-mls-pre	0.7	0.4	2.553	2.695	2.629	2.725	2.364	2.332	2.573
qi-mls-pre	0.7	0.5	2.554	2.695	2.627	2.726	2.367	2.337	2.574
qi-mls-pre	0.7	0.6	2.548	2.695	2.626	2.727	2.321	2.344	2.576
qi-mls-pre	0.7	0.7	2.551	2.695	2.626	2.728	2.325	2.353	2.578
qi-mls-pre	0.7	0.8	2.555	2.695	2.627	2.729	2.333	2.367	2.582
qi-mls-pre	0.7	0.9	2.565	2.695	2.629	2.733	2.351	2.394	2.59
qi-mls-pre	0.9	—	2.73	2.754	2.731	2.834	2.614	2.7	2.743
qi-mls-pre	0.9	0.1	2.555	2.699	2.64	2.728	2.362	2.329	2.573
qi-mls-pre	0.9	0.2	2.553	2.697	2.635	2.726	2.362	2.328	2.571
qi-mls-pre	0.9	0.3	2.553	2.696	2.632	2.725	2.362	2.329	2.572
qi-mls-pre	0.9	0.4	2.553	2.695	2.629	2.725	2.364	2.332	2.573
qi-mls-pre	0.9	0.5	2.554	2.695	2.627	2.726	2.367	2.337	2.574
qi-mls-pre	0.9	0.6	2.548	2.695	2.626	2.727	2.321	2.344	2.576
qi-mls-pre	0.9	0.7	2.551	2.695	2.626	2.728	2.325	2.353	2.578
qi-mls-pre	0.9	0.8	2.555	2.695	2.627	2.729	2.333	2.367	2.582
qi-mls-pre	0.9	0.9	2.565	2.695	2.629	2.733	2.351	2.394	2.59
qi-mls-pre-ng	0.1	—	2.327	2.673	2.451	2.351	1.956	2.167	2.362
qi-mls-pre-ng	0.1	0.1	2.254	2.664	2.413	2.285	1.903	1.979	2.279
qi-mls-pre-ng	0.1	0.2	2.266	2.656	2.409	2.28	1.903	2.077	2.274
qi-mls-pre-ng	0.1	0.3	2.259	2.656	2.399	2.286	1.901	2.038	2.272
qi-mls-pre-ng	0.1	0.4	2.258	2.654	2.397	2.278	1.893	2.055	2.271
qi-mls-pre-ng	0.1	0.5	2.258	2.654	2.396	2.278	1.896	2.055	2.271
qi-mls-pre-ng	0.1	0.6	2.259	2.656	2.397	2.278	1.896	2.056	2.271
qi-mls-pre-ng	0.1	0.7	2.263	2.656	2.398	2.279	1.917	2.056	2.271
qi-mls-pre-ng	0.1	0.8	2.261	2.655	2.4	2.282	1.91	2.053	2.267
qi-mls-pre-ng	0.1	0.9	2.282	2.648	2.415	2.285	1.995	2.047	2.299
qi-mls-pre-ng	0.3	—	2.079	2.025	1.951	2.208	2.058	2.128	2.103
qi-mls-pre-ng	0.3	0.1	1.965	1.962	1.924	2.158	1.818	1.898	2.031
qi-mls-pre-ng	0.3	0.2	1.959	1.959	1.917	2.154	1.813	1.885	2.026
qi-mls-pre-ng	0.3	0.3	1.957	1.96	1.913	2.155	1.806	1.883	2.026
qi-mls-pre-ng	0.3	0.4	1.954	1.962	1.911	2.156	1.798	1.874	2.025
qi-mls-pre-ng	0.3	0.5	1.956	1.961	1.911	2.156	1.805	1.875	2.025
qi-mls-pre-ng	0.3	0.6	1.956	1.962	1.911	2.157	1.805	1.875	2.025
qi-mls-pre-ng	0.3	0.7	1.961	1.962	1.911	2.157	1.834	1.875	2.025
qi-mls-pre-ng	0.3	0.8	1.96	1.962	1.912	2.16	1.828	1.876	2.021
qi-mls-pre-ng	0.3	0.9	1.959	1.963	1.915	2.161	1.849	1.866	2.001
qi-mls-pre-ng	0.5	—	2.079	2.025	1.951	2.208	2.058	2.128	2.103
qi-mls-pre-ng	0.5	0.1	1.965	1.962	1.924	2.158	1.818	1.898	2.031
qi-mls-pre-ng	0.5	0.2	1.959	1.959	1.917	2.154	1.813	1.885	2.027
qi-mls-pre-ng	0.5	0.3	1.957	1.96	1.913	2.155	1.807	1.883	2.026
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 22 (CREMI score on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls-pre-ng	0.5	0.4	1.955	1.962	1.911	2.156	1.799	1.874	2.025
qi-mls-pre-ng	0.5	0.5	1.956	1.961	1.911	2.156	1.805	1.875	2.025
qi-mls-pre-ng	0.5	0.6	1.956	1.962	1.911	2.157	1.805	1.875	2.025
qi-mls-pre-ng	0.5	0.7	1.961	1.962	1.911	2.157	1.834	1.875	2.026
qi-mls-pre-ng	0.5	0.8	1.96	1.962	1.912	2.16	1.829	1.876	2.021
qi-mls-pre-ng	0.5	0.9	1.959	1.963	1.915	2.161	1.849	1.866	2.001
qi-mls-pre-ng	0.7	—	2.079	2.025	1.951	2.208	2.058	2.128	2.103
qi-mls-pre-ng	0.7	0.1	1.965	1.962	1.924	2.158	1.818	1.898	2.031
qi-mls-pre-ng	0.7	0.2	1.959	1.959	1.917	2.154	1.813	1.885	2.027
qi-mls-pre-ng	0.7	0.3	1.957	1.96	1.913	2.155	1.807	1.883	2.026
qi-mls-pre-ng	0.7	0.4	1.955	1.962	1.911	2.156	1.799	1.874	2.025
qi-mls-pre-ng	0.7	0.5	1.956	1.961	1.911	2.156	1.805	1.875	2.025
qi-mls-pre-ng	0.7	0.6	1.956	1.962	1.911	2.157	1.805	1.875	2.025
qi-mls-pre-ng	0.7	0.7	1.961	1.962	1.911	2.157	1.834	1.875	2.026
qi-mls-pre-ng	0.7	0.8	1.96	1.962	1.912	2.16	1.829	1.876	2.021
qi-mls-pre-ng	0.7	0.9	1.959	1.963	1.915	2.161	1.849	1.866	2.001
qi-mls-pre-ng	0.9	—	2.079	2.025	1.951	2.208	2.058	2.128	2.103
qi-mls-pre-ng	0.9	0.1	1.965	1.962	1.924	2.158	1.818	1.898	2.031
qi-mls-pre-ng	0.9	0.2	1.959	1.959	1.917	2.154	1.813	1.885	2.027
qi-mls-pre-ng	0.9	0.3	1.957	1.96	1.913	2.155	1.807	1.883	2.026
qi-mls-pre-ng	0.9	0.4	1.955	1.962	1.911	2.156	1.799	1.874	2.025
qi-mls-pre-ng	0.9	0.5	1.956	1.961	1.911	2.156	1.805	1.875	2.025
qi-mls-pre-ng	0.9	0.6	1.956	1.962	1.911	2.157	1.805	1.875	2.025
qi-mls-pre-ng	0.9	0.7	1.961	1.962	1.911	2.157	1.834	1.875	2.026
qi-mls-pre-ng	0.9	0.8	1.96	1.962	1.912	2.16	1.829	1.876	2.021
qi-mls-pre-ng	0.9	0.9	1.959	1.963	1.915	2.161	1.849	1.866	2.001
qi-mls	0.1	—	0.852	0.555	0.482	1.061	0.872	1.211	0.933
qi-mls	0.1	0.1	0.659	0.505	0.439	1.007	0.481	0.842	0.682
qi-mls	0.1	0.2	0.652	0.502	0.432	1.005	0.471	0.828	0.674
qi-mls	0.1	0.3	0.65	0.503	0.435	1.004	0.465	0.822	0.669
qi-mls	0.1	0.4	0.647	0.505	0.424	1.004	0.464	0.82	0.667
qi-mls	0.1	0.5	0.647	0.504	0.422	1.003	0.462	0.827	0.665
qi-mls	0.1	0.6	0.647	0.504	0.422	1.003	0.46	0.83	0.665
qi-mls	0.1	0.7	0.648	0.506	0.422	1.004	0.46	0.832	0.665
qi-mls	0.1	0.8	0.651	0.508	0.421	1.005	0.461	0.845	0.666
qi-mls	0.1	0.9	0.657	0.512	0.425	1.008	0.467	0.86	0.67
qi-mls	0.3	—	1.477	1.476	1.227	1.703	1.141	1.773	1.54
qi-mls	0.3	0.1	1.344	1.433	1.195	1.663	0.913	1.456	1.407
qi-mls	0.3	0.2	1.344	1.435	1.193	1.663	0.913	1.458	1.405
qi-mls	0.3	0.3	1.345	1.436	1.189	1.663	0.914	1.461	1.403
qi-mls	0.3	0.4	1.346	1.438	1.19	1.664	0.917	1.462	1.404
qi-mls	0.3	0.5	1.347	1.439	1.189	1.665	0.916	1.463	1.407
qi-mls	0.3	0.6	1.351	1.444	1.19	1.666	0.929	1.467	1.41
qi-mls	0.3	0.7	1.354	1.448	1.19	1.668	0.934	1.47	1.412
qi-mls	0.3	0.8	1.357	1.453	1.192	1.67	0.936	1.475	1.416
qi-mls	0.3	0.9	1.365	1.464	1.197	1.674	0.943	1.488	1.422
qi-mls	0.5	—	3.306	3.285	3.401	3.394	3.47	3.061	3.226
qi-mls	0.5	0.1	3.291	3.287	3.416	3.388	3.398	3.015	3.239
qi-mls	0.5	0.2	3.289	3.286	3.415	3.387	3.397	3.012	3.238
qi-mls	0.5	0.3	3.288	3.285	3.414	3.386	3.396	3.011	3.236
qi-mls	0.5	0.4	3.288	3.285	3.413	3.386	3.396	3.01	3.236
qi-mls	0.5	0.5	3.287	3.285	3.412	3.386	3.395	3.01	3.235
qi-mls	0.5	0.6	3.287	3.285	3.412	3.386	3.395	3.009	3.235
qi-mls	0.5	0.7	3.287	3.285	3.412	3.387	3.395	3.01	3.235
qi-mls	0.5	0.8	3.288	3.285	3.412	3.387	3.395	3.011	3.235
qi-mls	0.5	0.9	3.289	3.285	3.414	3.389	3.397	3.015	3.236
qi-mls	0.7	—	3.306	3.285	3.401	3.394	3.47	3.061	3.226
qi-mls	0.7	0.1	3.291	3.287	3.416	3.388	3.398	3.015	3.239
qi-mls	0.7	0.2	3.289	3.286	3.415	3.387	3.397	3.012	3.238
qi-mls	0.7	0.3	3.288	3.285	3.414	3.386	3.396	3.011	3.236
qi-mls	0.7	0.4	3.288	3.285	3.413	3.386	3.396	3.01	3.236
qi-mls	0.7	0.5	3.287	3.285	3.412	3.386	3.395	3.01	3.235
qi-mls	0.7	0.6	3.287	3.285	3.412	3.386	3.395	3.009	3.235
qi-mls	0.7	0.7	3.287	3.285	3.412	3.387	3.395	3.01	3.235
qi-mls	0.7	0.8	3.288	3.285	3.412	3.387	3.395	3.011	3.235
qi-mls	0.7	0.9	3.289	3.285	3.414	3.389	3.397	3.015	3.236
qi-mls	0.9	—	3.306	3.285	3.401	3.394	3.47	3.061	3.226
Architecture	t_m	t_g	∅	A	B	C	0	1	2

Continued on next page...

...continued: Table 22 (CREMI score on 100%)

Architecture	t_m	t_g	∅	A	B	C	0	1	2
qi-mls	0.9	0.1	3.291	3.287	3.416	3.388	3.398	3.015	3.239
qi-mls	0.9	0.2	3.289	3.286	3.415	3.387	3.397	3.012	3.238
qi-mls	0.9	0.3	3.288	3.285	3.414	3.386	3.396	3.011	3.236
qi-mls	0.9	0.4	3.288	3.285	3.413	3.386	3.396	3.01	3.236
qi-mls	0.9	0.5	3.287	3.285	3.412	3.386	3.395	3.01	3.235
qi-mls	0.9	0.6	3.287	3.285	3.412	3.386	3.395	3.009	3.235
qi-mls	0.9	0.7	3.287	3.285	3.412	3.387	3.395	3.01	3.235
qi-mls	0.9	0.8	3.288	3.285	3.412	3.387	3.395	3.011	3.235
qi-mls	0.9	0.9	3.289	3.285	3.414	3.389	3.397	3.015	3.236
qi-mls-ng	0.1	—	2.472	2.748	2.558	2.544	2.183	2.376	2.422
qi-mls-ng	0.1	0.1	2.471	2.743	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.1	0.2	2.471	2.743	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.1	0.3	2.469	2.743	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.1	0.4	2.469	2.743	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.1	0.5	2.467	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.1	0.6	2.467	2.731	2.558	2.544	2.174	2.367	2.426
qi-mls-ng	0.1	0.7	2.466	2.731	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.1	0.8	2.466	2.731	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.1	0.9	2.469	2.745	2.558	2.544	2.174	2.368	2.424
qi-mls-ng	0.3	—	2.47	2.735	2.558	2.544	2.183	2.376	2.422
qi-mls-ng	0.3	0.1	2.469	2.731	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.3	0.2	2.469	2.731	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.3	0.3	2.466	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.3	0.4	2.467	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.3	0.5	2.466	2.727	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.3	0.6	2.466	2.727	2.558	2.544	2.174	2.367	2.426
qi-mls-ng	0.3	0.7	2.466	2.727	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.3	0.8	2.466	2.727	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.3	0.9	2.466	2.728	2.558	2.544	2.174	2.368	2.424
qi-mls-ng	0.5	—	2.47	2.735	2.558	2.544	2.183	2.376	2.422
qi-mls-ng	0.5	0.1	2.469	2.731	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.5	0.2	2.469	2.731	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.5	0.3	2.466	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.5	0.4	2.467	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.5	0.5	2.466	2.727	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.5	0.6	2.466	2.727	2.558	2.544	2.174	2.367	2.426
qi-mls-ng	0.5	0.7	2.466	2.727	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.5	0.8	2.466	2.727	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.5	0.9	2.466	2.728	2.558	2.544	2.174	2.368	2.424
qi-mls-ng	0.7	—	2.47	2.735	2.558	2.544	2.183	2.376	2.422
qi-mls-ng	0.7	0.1	2.469	2.731	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.7	0.2	2.469	2.731	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.7	0.3	2.466	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.7	0.4	2.467	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.7	0.5	2.466	2.727	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.7	0.6	2.466	2.727	2.558	2.544	2.174	2.367	2.426
qi-mls-ng	0.7	0.7	2.466	2.727	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.7	0.8	2.466	2.727	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.7	0.9	2.466	2.728	2.558	2.544	2.174	2.368	2.424
qi-mls-ng	0.9	—	2.47	2.735	2.558	2.544	2.183	2.376	2.422
qi-mls-ng	0.9	0.1	2.469	2.731	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.9	0.2	2.469	2.731	2.558	2.544	2.173	2.367	2.44
qi-mls-ng	0.9	0.3	2.466	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.9	0.4	2.467	2.731	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.9	0.5	2.466	2.727	2.558	2.544	2.173	2.367	2.426
qi-mls-ng	0.9	0.6	2.466	2.727	2.558	2.544	2.174	2.367	2.426
qi-mls-ng	0.9	0.7	2.466	2.727	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.9	0.8	2.466	2.727	2.558	2.544	2.174	2.367	2.424
qi-mls-ng	0.9	0.9	2.466	2.728	2.558	2.544	2.174	2.368	2.424
Architecture	t_m	t_g	∅	A	B	C	0	1	2

C

SOFTWARE

C.1 PAINTERA

C.1.1 *Data Conversion*

This example shows how to convert the `/volumes/raw` and `/volumes/labels/neuron_ids` data sets of padded CREMI sample A[1] into multi-scale, Paintera compatible datasets. First, install the paintera-conversion-helper tool (section 4.1.5.5). In the following, `$HOME` refers to the user's home directory. Assuming that the padded sample A was downloaded into the `$HOME/Downloads` directory, the command

Listing C.1: Conversion of HDF5 data into Paintera-friendly format.

```
paintera-conversion-helper \
    -d $HOME/Downloads/sample_A_padded_20160501.hdf,volumes/raw,raw \
    -d $HOME/Downloads/sample_A_padded_20160501.hdf,volumes/labels/neuron_ids,label \
    -b 64,64,64 \
    -s 2,2,1 2,2,1 2,2,1 2,2,2 2,2,2 2,2,2 \
    -m -1 -1 5 4 3 2 2 1 \
    --revert \
    --label-block-lookup-backend-n5=10000 \
    --outputN5=$HOME/data/sample_A_padded_20160501.n5
```

creates a new N5 container `$HOME/data/sample_A_padded_20160501.n5` with the converted datasets. During conversion, you can track the progress by opening the Spark Jobs status page in a web browser, *e.g.* `http://localhost:4040/jobs` for local Spark Jobs. Then, you can either open the datasets through the Paintera UI or load a Paintera project, *e.g.* `paintera $HOME/data/project.n5` with the following attributes.json (Replace `$HOME` appropriately):

Listing C.2: Example Paintera project

```
{
  "n5": "2.0.2",
  "paintera": {
    "sourceInfo": {
      "sources": [
        {
          "type": "org.janelia.saalfeldlab.paintera.state.RawSourceState",
          "state": {
            "composite": {},
            "converter": {
              "alpha": 1.0,
              "color": "#FFFFFF",
              "min": 0.0,
              "max": 255.0
            },
            "compositeType": "org.janelia.saalfeldlab.paintera.composition.CompositeCopy",
            "converterType": "net.imglib2.converter.ARGBColorConverter$InvertingImp1",
            "interpolation": "NEARESTNEIGHBOR",
```

[1] https://cremi.org/static/data/sample_A_padded_20160501.hdf

```
            "isVisible": true,
            "sourceType": "org.janelia.saalfeldlab.paintera.data.n5.N5DataSource",
            "source": {
              "metaType": "org.janelia.saalfeldlab.paintera.data.n5.N5FSMeta",
              "meta": {
                "n5": "$HOME/data/sample_A_padded_20160501.n5",
                "dataset": "/volumes/raw"
              },
              "transform": [
                4.0,
                0.0,
                0.0,
                0.0,
                0.0,
                4.0,
                0.0,
                0.0,
                0.0,
                0.0,
                40.0,
                0.0
              ]
            },
            "name": "raw",
            "dependsOn": [],
            "axisOrder": "XYZ"
          }
        },
        {
          "type": "org.janelia.saalfeldlab.paintera.state.LabelSourceState",
          "state": {
            "composite": {},
            "converter": {
              "converterType": "org.janelia.saalfeldlab.paintera.stream.
                   HighlightingStreamConverterLabelMultisetType",
              "streamType": "org.janelia.saalfeldlab.paintera.stream.
                   ModalGoldenAngleSaturatedHighlightingARGBStream",
              "seed": 1
            },
            "compositeType": "org.janelia.saalfeldlab.paintera.composition.
                ARGBCompositeAlphaYCbCr",
            "converterType": "org.janelia.saalfeldlab.paintera.stream.
                HighlightingStreamConverterLabelMultisetType",
            "interpolation": "NEARESTNEIGHBOR",
            "isVisible": true,
            "sourceType": "org.janelia.saalfeldlab.paintera.data.mask.MaskedSource",
            "source": {
              "source": {
                "metaType": "org.janelia.saalfeldlab.paintera.data.n5.N5FSMeta",
                "meta": {
                  "n5": "$HOME/data/sample_A_padded_20160501.n5",
                  "dataset": "/volumes/labels/neuron_ids"
                },
                "transform": [
                  4.0,
                  0.0,
                  0.0,
                  3644.0,
                  0.0,
                  4.0,
                  0.0,
                  3644.0,
                  0.0,
                  0.0,
                  40.0,
                  1520.0
```

```
            ]
          },
          "sourceClass": "org.janelia.saalfeldlab.paintera.data.n5.N5DataSource",
          "persistCanvasClass": "org.janelia.saalfeldlab.paintera.data.n5.CommitCanvasN5",
          "persistCanvas": {
            "class": "org.janelia.saalfeldlab.paintera.data.n5.N5FSMeta",
            "data": {
              "n5": "$HOME/data/sample_A_padded_20160501.n5",
              "dataset": "/volumes/labels/neuron_ids"
            }
          }
        },
        "name": "neuron_ids",
        "dependsOn": [],
        "axisOrder": "XYZ",
        "selectedIds": {
          "lastSelection": -2,
          "activeIds": []
        },
        "assignment": {
          "type": "org.janelia.saalfeldlab.paintera.control.assignment.
              FragmentSegmentAssignmentOnlyLocal",
          "data": {
            "actions": [],
            "persister": {
              "type": "org.janelia.saalfeldlab.util.n5.N5
                  FragmentSegmentAssignmentPersister",
              "data": {
                "N5": {
                  "type": "org.janelia.saalfeldlab.paintera.data.n5.N5FSMeta",
                  "data": {
                    "n5": "$HOME/data/sample_A_padded_20160501.n5",
                    "dataset": "/volumes/labels/neuron_ids/fragment-segment-assignment"
                  }
                }
              }
            },
            "initialLut": {
              "type": "org.janelia.saalfeldlab.util.n5.N5
                  FragmentSegmentAssignmentInitialLut",
              "data": {
                "N5": {
                  "type": "org.janelia.saalfeldlab.paintera.data.n5.N5FSMeta",
                  "data": {
                    "n5": "$HOME/data/sample_A_padded_20160501.n5",
                    "dataset": "/volumes/labels/neuron_ids/fragment-segment-assignment"
                  }
                }
              }
            }
          }
        },
        "lockedSegments": [],
        "meshSettings": {
          "globalSettings": {
            "numScaleLevels": 7,
            "scaleLevel": 6,
            "simplificationIterations": 0,
            "smoothingLambda": 0.5,
            "smoothingIterations": 3,
            "opacity": 1.0,
            "inflate": 1.0,
            "isVisible": true,
            "drawMode": "FILL",
            "cullFace": "FRONT"
          },
```

```
            "meshSettings": []
          },
          "labelBlockMapping": {
            "type": "n5-filesystem",
            "root": "$HOME/data/sample_A_padded_20160501.n5",
            "scaleDatasetPattern": "volumes/labels/neuron_ids/label-to-block-mapping/s%d"
          },
          "idService": {
            "type": "org.janelia.saalfeldlab.paintera.id.N5IdService",
            "data": {
              "N5": {
                "type": "org.janelia.saalfeldlab.paintera.data.n5.N5FSMeta",
                "data": {
                  "n5": "$HOME/data/sample_A_padded_20160501.n5",
                  "dataset": "/volumes/labels/neuron_ids"
                }
              }
            }
          }
        }
      ],
      "currentSourceIndex": 0,
      "numSources": 2
    },
    "globalTransform": [
      0.06512006512006512,
      0.0,
      0.0,
      -399.96743996743993,
      0.0,
      0.06512006512006512,
      0.0,
      -399.96743996743993,
      0.0,
      0.0,
      0.06512006512006512,
      -259.17785917785915
    ],
    "windowProperties": {
      "width": 1600,
      "height": 800
    },
    "gridConstraints": {
      "previousFirstRowHeight": 50.0,
      "previousFirstColumnWidth": 50.0,
      "isFullScreen": false,
      "firstRowHeight": 50.0,
      "firstColumnWidth": 50.0
    },
    "crosshairConfig": {
      "onFocusColor": "#FF0088FF",
      "offFocusColor": "#FFFFFFFF",
      "isVisible": true
    },
    "orthoSliceConfig": {
      "enabled": true,
      "showTopLeft": true,
      "showTopRight": true,
      "showBottomLeft": true,
      "delayInNanoSeconds": 200
    },
    "navigationConfig": {
      "allowRotations": true,
      "buttonRotationSpeeds": {
        "slow": 0.5,
```

```
        "regular": 5.0,
        "fast": 45.0
      }
    },
    "viewer3DConfig": {
      "areMeshesEnabled": true
    },
    "screenScalesConfig": {
      "scales": [
        1.0,
        0.5,
        0.25
      ]
    }
  }
}
```

C.1.2 *Extensions — Complete Example*

The extension example provided in section 4.1.7 can be implemented as a Maven project with a single Java file:

```
.
├── pom.xml
└── src
    └── main
        └── java
            └── my
                └── group
                    └── FeatureSourceState.java
```

with `pom.xml`:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <parent>
                <groupId>org.scijava</groupId>
                <artifactId>pom-scijava</artifactId>
                <version>25.0.0</version>
        </parent>

        <properties>
                <enforcer.skip>true</enforcer.skip>
        </properties>

        <groupId>my.group</groupId>
        <artifactId>my.artifact</artifactId>
        <version>0.1.0-SNAPSHOT</version>

        <repositories>
                <repository>
                        <id>imagej.public</id>
                        <url>http://maven.imagej.net/content/groups/public</url>
                </repository>
        </repositories>

```

```
        <dependencies>
                <dependency>
                        <groupId>org.janelia.saalfeldlab</groupId>
                        <artifactId>paintera</artifactId>
                        <version>0.11.1</version>
                </dependency>
    </dependencies>
</project>
```

The single java source file contains all the necessary code for the gradient feature extension including an additional entry for the opener menu and support for serialization:

```
package my.group;

import bdv.util.volatiles.VolatileViews;
import bdv.viewer.Interpolation;
import com.google.gson.JsonDeserializationContext;
import com.google.gson.JsonDeserializer;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParseException;
import com.google.gson.JsonSerializationContext;
import com.google.gson.JsonSerializer;
import com.google.gson.annotations.Expose;
import javafx.beans.InvalidationListener;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.control.ComboBox;
import net.imglib2.RandomAccessible;
import net.imglib2.RandomAccessibleInterval;
import net.imglib2.algorithm.gradient.PartialDerivative;
import net.imglib2.cache.img.CellLoader;
import net.imglib2.cache.img.DiskCachedCellImgFactory;
import net.imglib2.cache.img.DiskCachedCellImgOptions;
import net.imglib2.converter.ARGBColorConverter;
import net.imglib2.converter.Converters;
import net.imglib2.interpolation.InterpolatorFactory;
import net.imglib2.interpolation.randomaccess.NLinearInterpolatorFactory;
import net.imglib2.interpolation.randomaccess.NearestNeighborInterpolatorFactory;
import net.imglib2.loops.LoopBuilder;
import net.imglib2.realtransform.AffineTransform3D;
import net.imglib2.type.numeric.NumericType;
import net.imglib2.type.numeric.RealType;
import net.imglib2.type.numeric.real.DoubleType;
import net.imglib2.type.volatiles.VolatileDoubleType;
import net.imglib2.view.Views;
import org.janelia.saalfeldlab.paintera.PainteraBaseView;
import org.janelia.saalfeldlab.paintera.composition.ARGBCompositeAlphaAdd;
import org.janelia.saalfeldlab.paintera.data.DataSource;
import org.janelia.saalfeldlab.paintera.data.RandomAccessibleIntervalDataSource;
import org.janelia.saalfeldlab.paintera.serialization.StatefulSerializer;
import org.janelia.saalfeldlab.paintera.state.MinimalSourceState;
import org.janelia.saalfeldlab.paintera.state.SourceState;
import org.janelia.saalfeldlab.paintera.ui.PainteraAlerts;
import org.janelia.saalfeldlab.paintera.ui.opendialog.menu.OpenDialogMenuEntry;
import org.janelia.saalfeldlab.util.Colors;
import org.scijava.plugin.Plugin;

import java.lang.reflect.Type;
import java.nio.file.Paths;
import java.util.List;
import java.util.Objects;
import java.util.Optional;
```

```
54 import java.util.function.BiConsumer;
55 import java.util.function.Function;
56 import java.util.function.IntFunction;
57 import java.util.function.Predicate;
58 import java.util.function.Supplier;
59 import java.util.function.ToIntFunction;
60 import java.util.stream.Collectors;
61 import java.util.stream.IntStream;
62 import java.util.stream.Stream;
63
64 import static org.janelia.saalfeldlab.paintera.serialization.SerializationHelpers.
       deserializeFromClassInfo;
65 import static org.janelia.saalfeldlab.paintera.serialization.SerializationHelpers.
       serializeWithClassInfo;
66 import static org.janelia.saalfeldlab.paintera.serialization.sourcestate.
       SourceStateSerialization.DEPENDS_ON_KEY;
67 import static org.janelia.saalfeldlab.paintera.serialization.sourcestate.
       SourceStateSerialization.INTERPOLATION_KEY;
68 import static org.janelia.saalfeldlab.paintera.serialization.sourcestate.
       SourceStateSerialization.IS_VISIBLE_KEY;
69 import static org.janelia.saalfeldlab.paintera.serialization.sourcestate.
       SourceStateSerialization.NAME_KEY;
70
71 public class FeatureSourceState extends MinimalSourceState<
72         DoubleType,
73         VolatileDoubleType,
74         DataSource<DoubleType, VolatileDoubleType>,
75         ARGBColorConverter<VolatileDoubleType>> {
76
77     private interface Feature {
78         DataSource<DoubleType, VolatileDoubleType> featureSource(
79                 String cacheDir,
80                 SourceState<? extends RealType<?>, ?>... dependsOn);
81     }
82
83     private static class GradientFeature implements Feature {
84
85         @Expose
86         private final int dim;
87
88
89         public GradientFeature(final int dim) {
90             this.dim = dim;
91         }
92
93         @Override
94         public DataSource<DoubleType, VolatileDoubleType> featureSource(
95                 final String cacheDir,
96                 final SourceState<? extends RealType<?>, ?>... dependsOn) {
97             if (dependsOn.length != 1)
98                 throw new RuntimeException("Expected exactly one dependency but got " +
                        dependsOn.length);
99             final DataSource<? extends RealType<?>, ?> dataSource = dependsOn[0].getDataSource
                    ();
100            final int numLevels = dataSource.getNumMipmapLevels();
101            final AffineTransform3D[] tfs = IntStream
102                    .range(0, numLevels)
103                    .mapToObj(lvl -> { AffineTransform3D tf = new AffineTransform3D();
                            dataSource.getSourceTransform(0, lvl, tf); return tf;})
104                    .toArray(AffineTransform3D[]::new);
105
106            final RandomAccessibleInterval<DoubleType>[] data = new RandomAccessibleInterval[
                    numLevels];
107            final RandomAccessibleInterval<VolatileDoubleType>[] vdata = new
                    RandomAccessibleInterval[numLevels];
108
```

```
            final DiskCachedCellImgOptions options = DiskCachedCellImgOptions
                    .options()
                    .tempDirectory(Paths.get(cacheDir))
                    .tempDirectoryPrefix("gradient-")
                    .deleteCacheDirectoryOnExit(true)
                    .cellDimensions(32, 32, 32)
                    .volatileAccesses(true);


            for (int lvl = 0; lvl < numLevels; ++lvl) {
                final RandomAccessibleInterval<DoubleType> raw = Converters.convert(
                        dataSource.getDataSource(0, lvl),
                        (src, tgt) -> tgt.setReal(src.getRealDouble()),
                        new DoubleType());
                final int nDim = raw.numDimensions();
                final RandomAccessible<DoubleType> rawExtended = Views.extendBorder(raw);
                final DiskCachedCellImgFactory<DoubleType> factory = new
                     DiskCachedCellImgFactory<>(new DoubleType(), options);

                CellLoader<DoubleType> loader = img -> PartialDerivative.
                     gradientCentralDifference(rawExtended, img, dim);

                data[lvl] = factory.create(raw, loader, options);
                vdata[lvl] = VolatileViews.wrapAsVolatile(data[lvl]);
            }
            return new RandomAccessibleIntervalDataSource<>(
                    data,
                    vdata,
                    tfs,
                    () -> {
                    },
                    new InterpolationFunc<>(),
                    new InterpolationFunc<>(),
                    "");
        }
    }

    private static class MagnitudeFeature implements Feature {

            @Override
            public DataSource<DoubleType, VolatileDoubleType> featureSource(
                    final String cacheDir,
                    final SourceState<? extends RealType<?>, ?>... dependsOn) {
                // TODO check consistency of all sources, as long as it is called only
                     privately, do not care
                final DataSource<? extends RealType<?>, ?> dataSource = dependsOn[0].
                     getDataSource();
                final int numLevels = dataSource.getNumMipmapLevels();
                final AffineTransform3D[] tfs = IntStream
                        .range(0, numLevels)
                        .mapToObj(lvl -> { AffineTransform3D tf = new AffineTransform3D();
                             dataSource.getSourceTransform(0, lvl, tf); return tf;})
                        .toArray(AffineTransform3D[]::new);

                final RandomAccessibleInterval<DoubleType>[] data = new
                     RandomAccessibleInterval[numLevels];
                final RandomAccessibleInterval<VolatileDoubleType>[] vdata = new
                     RandomAccessibleInterval[numLevels];

                final DiskCachedCellImgOptions options = DiskCachedCellImgOptions
                        .options()
                        .tempDirectory(Paths.get(cacheDir))
                        .tempDirectoryPrefix("gradient-")
                        .deleteCacheDirectoryOnExit(true)
                        .cellDimensions(32, 32, 32)
                        .volatileAccesses(true);
```

```
168
169
170             for (int lvl = 0; lvl < numLevels; ++lvl) {
171                 final RandomAccessibleInterval<DoubleType> raw = Converters.convert(
172                         dataSource.getDataSource(0, lvl),
173                         (src, tgt) -> tgt.setReal(src.getRealDouble()),
174                         new DoubleType());
175                 final DiskCachedCellImgFactory<DoubleType> factory = new
                         DiskCachedCellImgFactory<>(new DoubleType(), options);
176                 final int flvl = lvl;
177
178                 CellLoader<DoubleType> loader = img -> {
179                     for (SourceState<? extends RealType<?>, ?> state : dependsOn) {
180                         LoopBuilder
181                                 .setImages(Views.interval(state.getDataSource().
                                         getDataSource(0, flvl), img), img)
182                                 .forEachPixel((src, tgt) -> tgt.setReal(tgt.getRealDouble
                                         () + src.getRealDouble() * src.getRealDouble()));
183                     }
184                     img.forEach(px -> px.setReal(Math.sqrt(px.getRealDouble())));
185                 };
186
187                 data[lvl] = factory.create(raw, loader, options);
188                 vdata[lvl] = VolatileViews.wrapAsVolatile(data[lvl]);
189             }
190             return new RandomAccessibleIntervalDataSource<>(
191                     data,
192                     vdata,
193                     tfs,
194                     () -> {
195                     },
196                     new InterpolationFunc<>(),
197                     new InterpolationFunc<>(),
198                     "");
199         }
200     }
201
202     private final Feature feature;
203
204     private FeatureSourceState(
205             final Feature feature,
206             final String name,
207             final String cacheDir,
208             SourceState<? extends RealType<?>, ?>... dependsOn) {
209         super(
210                 feature.featureSource(cacheDir, dependsOn),
211                 new ARGBColorConverter.InvertingImp1<   >(),
212                 new ARGBCompositeAlphaAdd(),
213                 name,
214                 dependsOn);
215         this.feature = feature;
216         converter().setMin(0.0);
217         converter().setMax(50.0);
218     }
219
220     private static class InterpolationFunc<D extends NumericType<D>>
221             implements Function<Interpolation, InterpolatorFactory<D, RandomAccessible<D>>> {
222
223         @Override
224         public InterpolatorFactory<D, RandomAccessible<D>> apply(Interpolation interpolation)
                 {
225             if (interpolation == Interpolation.NLINEAR)
226                 return new NLinearInterpolatorFactory<>();
227             else
228                 return new NearestNeighborInterpolatorFactory<>();
229         }
```

```
230     }
231
232     @Plugin(type = OpenDialogMenuEntry.class,
233             menuPath = "_Features>_Gradient Magnitude")
234     public static class MenuEntry implements OpenDialogMenuEntry {
235
236         @Override
237         public BiConsumer<PainteraBaseView, String> onAction() {
238             return (pbv, directory) -> {
239                 final Predicate<SourceState<?, ?>> isReal =
240                         state -> state
241                                 .getDataSource()
242                                 .getDataType() instanceof RealType<?>;
243                     final List<SourceState<? extends RealType<?>, ?>> sources =
244                             pbv
245                             .sourceInfo()
246                             .trackSources()
247                             .stream()
248                             .map(pbv.sourceInfo()::getState)
249                             .filter(isReal)
250                             .map(s -> (SourceState<? extends RealType<?>, ?>)s)
251                             .collect(Collectors.toList());
252                     final Alert alert = PainteraAlerts.alert(
253                             Alert.AlertType.CONFIRMATION,
254                             true);
255                     final ObservableList<SourceState<? extends RealType<?>, ?>> choices =
256                             FXCollections.observableArrayList(sources);
257                     final ComboBox<SourceState<? extends RealType<?>, ?>> comboBox =
258                             new ComboBox<>(choices);
259                     alert.getDialogPane().setContent(comboBox);
260                     final Optional<ButtonType> bt = alert.showAndWait();
261                     if (bt.filter(ButtonType.OK::equals).isPresent() && comboBox.getValue() !=
                              null) {
262                     final SourceState<? extends RealType<?>, ?> raw = comboBox.getValue();
263                     final int nDim = raw.getDataSource().getDataSource(0, 0).numDimensions();
264                     final FeatureSourceState[] gradients = IntStream
265                             .range(0, nDim)
266                             .mapToObj(dim -> new GradientFeature(dim))
267                             .map(feat -> new FeatureSourceState(
268                                     feat,
269                                     raw.nameProperty().getName() + "-gradient", directory, raw
                                          ))
270                             .toArray(FeatureSourceState[]::new);
271                     final FeatureSourceState magnitude = new FeatureSourceState(
272                             new MagnitudeFeature(),
273                             raw.nameProperty().getName() + "-gradient-magnitude",
274                             directory,
275                             gradients);
276                     gradients[0]
277                             .converter()
278                             .setColor(Colors.toARGBType("#ff0000"));
279                     gradients[1]
280                             .converter()
281                             .setColor(Colors.toARGBType("#00ff00"));
282                     gradients[2]
283                             .converter()
284                             .setColor(Colors.toARGBType("#0000ff"));
285                     Stream.of(gradients).forEach(pbv::addState);
286                     Stream.of(magnitude).forEach(pbv::addState);
287                 }
288             };
289         }
290     }
291
292     @Override
293     public void onAdd(PainteraBaseView paintera) {
```

```
294         InvalidationListener reqRep = obs -> paintera
295                 .orthogonalViews()
296                 .requestRepaint();
297         converter().minProperty().addListener(reqRep);
298         converter().maxProperty().addListener(reqRep);
299         converter().colorProperty().addListener(reqRep);
300         converter().alphaProperty().addListener(reqRep);
301     }
302
303 @Plugin(type = StatefulSerializer.SerializerAndDeserializer.class)
304 public static class SerializationFactory implements
305         StatefulSerializer.SerializerAndDeserializer<
306         FeatureSourceState,
307         Deserializer,
308         Serializer> {
309
310     @Override
311     public Deserializer createDeserializer(
312             StatefulSerializer.Arguments arguments,
313             Supplier<String> projectDirectory,
314             IntFunction<SourceState<?, ?>> dependencyFromIndex) {
315         return new Deserializer(
316                 dependencyFromIndex,
317                 projectDirectory.get());
318     }
319
320     @Override
321     public Serializer createSerializer(
322             Supplier<String> projectDirectory,
323             ToIntFunction<SourceState<?, ?>> stateToIndex) {
324         return new Serializer(stateToIndex);
325     }
326
327     @Override
328     public Class<FeatureSourceState> getTargetClass() {
329         return FeatureSourceState.class;
330     }
331 }
332
333 private static class Serializer implements JsonSerializer<FeatureSourceState> {
334
335     private final ToIntFunction<SourceState<?, ?>> sourceToIndex;
336
337     private Serializer(final ToIntFunction<SourceState<?, ?>> sourceToIndex) {
338         this.sourceToIndex = sourceToIndex;
339     }
340
341     @Override
342     public JsonElement serialize(
343             FeatureSourceState src,
344             Type typeOfSrc,
345             JsonSerializationContext context) {
346         final JsonObject map = new JsonObject();
347         map.add("composite", serializeWithClassInfo(
348                 src.compositeProperty().get(),
349                 context));
350         map.add("converter", serializeWithClassInfo(
351                 src.converter(),
352                 context));
353         map.add("feature", serializeWithClassInfo(
354                 src.feature,
355                 context));
356         map.add(INTERPOLATION_KEY, context.serialize(
357                 src.interpolationProperty().get(),
358                 Interpolation.class));
359         map.addProperty(
```

```
                IS_VISIBLE_KEY,
                src.isVisibleProperty().get());
        map.addProperty(NAME_KEY, src.nameProperty().get());
        map.add(DEPENDS_ON_KEY, context.serialize(Stream
                .of(src.dependsOn())
                .mapToInt(sourceToIndex)
                .toArray()));
        return map;
    }
}

    private static class Deserializer implements JsonDeserializer<FeatureSourceState> {
        private final IntFunction<SourceState<?, ?>> dependencyFromIndex;

        private final String cacheDir;

        private Deserializer(
                final IntFunction<SourceState<?, ?>> dependencyFromIndex,
                final String cacheDir) {
            this.dependencyFromIndex = dependencyFromIndex;
            this.cacheDir = cacheDir;
        }

        @Override
        public FeatureSourceState deserialize(
                JsonElement json,
                Type typeOfT,
                JsonDeserializationContext context) throws JsonParseException {
            final JsonObject map = json.getAsJsonObject();
            try {
                final SourceState<? extends RealType<?>, ?>[] dependsOn =
                        IntStream
                        .of(context.deserialize(map.get(DEPENDS_ON_KEY), int[].class))
                        .mapToObj(dependencyFromIndex)
                        .toArray(SourceState[]::new);
                if (Stream.of(dependsOn).anyMatch(Objects::isNull))
                    return null;
                final FeatureSourceState fs = new FeatureSourceState(
                        deserializeFromClassInfo(
                                map.getAsJsonObject("feature"),
                                context),
                        map.get(NAME_KEY).getAsString(),
                        cacheDir,
                        dependsOn);
                final ARGBColorConverter<VolatileDoubleType> converter =
                        deserializeFromClassInfo(
                                map.getAsJsonObject("converter"),
                                context);
                fs.converter().setColor(converter.getColor());
                fs.converter().setMin(converter.getMin());
                fs.converter().setMax(converter.getMax());
                fs.converter().alphaProperty().set(converter.alphaProperty().get());
                fs.compositeProperty().set(deserializeFromClassInfo(
                        map.getAsJsonObject("composite"),
                        context));
                fs.interpolationProperty().set(context.deserialize(
                        map.get(INTERPOLATION_KEY),
                        Interpolation.class));
                fs.isVisibleProperty().set(map.get(IS_VISIBLE_KEY).getAsBoolean());
                return fs;
            } catch (ClassNotFoundException e) {
                throw new JsonParseException(e);
            }
        }
    }
}
```

Paintera can be executed with this extension simply by installing the project into the local maven repository

```
mvn clean install
```

and adding the project as additional endpoint to the Paintera command:

```
paintera \
    --additional-endpoints \
    my.group:my.artifact:0.1.0-SNAPSHOT \
    --
```

The installation step can be skipped if an extension is available through remote maven repositories.