DTU Compute
Department of Applied Mathematics and Computer Science

Axon segmentation using 3D Convolutional Neural Networks

MSc Thesis

Miren Lur Barquin Torre

Axon segmentation using 3D Convolutional Neural Networks
Segmentering af axoner med 3D convolutional neurale networks

MSc Thesis
June, 2023

By
Miren Lur Barquin Torre

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby, Denmark
www.compute.dtu.dk
ISSN: [0000-0000] (electronic version)
ISBN: [000-00-0000-000-0] (electronic version)

ISSN: [0000-0000] (printed version)
ISBN: [000-00-0000-000-0] (printed version)

Preface

This thesis has been prepared over five months at the Department of Applied Mathematics and Computer Science of the Technical University of Denmark (DTU Compute), in partial fulfillment for the degree Master of Science in Engineering, MSc Eng., on Mathematical Modelling and Computation. The research presented in this thesis was conducted from February 2023 to June 2023.

The project was carried out under the primary supervision of Professor Tim B. Dyrby from DTU Compute and the Danish Research Centre for Magnetic Resonance (DRCMR), Copenhagen University Hospital Hvidovre.

The purpose of this dissertation is to explore and analyze the three-dimensional microstructure of the brain white matter by segmenting axons using 3D Convolutional Neural Networks in 3D synchrotron X-ray nano-holotomography images, aiming to overcome the current limitations of diffusion magnetic resonance imaging methods for measuring axon diameter.

It is assumed that the reader has a basic knowledge in the areas of Machine Learning and Computer Vision.

Miren Lur Barquin Torre - s210289

.....

Signature

30th June 2023

Abstract

Axons are long nerve fibers that facilitate signal communication between different functional brain sections and constitute a significant portion of the brain's white matter. Their diameter and myelin sheath thickness play a crucial role in determining the speed of signal propagation within the brain network. While commonly modelled as straight cylinders, detailed studies using light microscopy and electron microscopy have revealed variations in diameter, shape, and trajectory along axons. The presence of local microstructural obstacles in the extra-cellular space such as blood vessels, vacuoles, and cell clusters, can alter axonal trajectories and bias axon diameter measurements. Moreover, the diameter of axons, which directly influences conduction velocity, varies based on the origin and target of the axon and typically ranges within a few micrometers. Certain neurodegenerative diseases specifically affect axons of particular sizes, making axon diameter a valuable biomarker for diagnosing various medical conditions.

Currently, diffusion magnetic resonance imaging (MRI) is the only method available for estimating axonal diameter in the living brain. However, it has been found to overestimate axonal diameters compared to measurements obtained through light microscopy and electron microscopy. Moreover, MRI voxels for in vivo studies are on a millimeter scale, whereas axon diameters in the brain are on a micrometer scale.

This dissertation aims to validate the white matter anatomy and the biophysical models aforementioned by conducting a 3D analysis of axonal and extra-axonal structures. Specifically, the focus is on testing the validity of modelling axons as cylinders and exploring the morphological characteristics of axonal structures. To achieve this, the dissertation focuses on the segmentation of 3D axonal structures using various 3D convolutional neural network methods based on a U-Net network architecture. The data utilized in this study consists of a volumetric image sample extracted from the splenium of the vervet monkey brain via synchrotron XNH imaging techniques. The ground truth label map includes a manual segmentation of 54-axons, as well as a morphology-based semi-automatic segmentation of blood vessels, cell clusters, and vacuoles, that will facilitate the evaluation of the experiment performance. By implementing appropriate data preprocessing techniques and selecting a masked loss function, the training process is enhanced, resulting in the ability to capture a denser population of axons, including those with smaller diameters.

The primary research objective of this dissertation is to achieve a more comprehensive three-dimensional segmentation of axonal and extra-axonal structures, with a particular emphasis on accurately capturing axons with varying diameters.

Acknowledgements

Miren Lur Barquin Torre, MSc Mathematical Modelling and Computation, DTU
Thesis author

Tim Bjørn Dyrby, Professor, DTU Compute/Hvidovre Hospital
Thesis supervisor

Mahsa Amirrashedibonab, Postdoc, DTU Compute
Thesis co-supervisor

Abbreviations

AD Axon Diameter
ADD Axon Diameter Distribution
AP Action Potentials
CC Corpus Callosum
CE Cross Entropy
CNN Convolutional Neural Network
CNS Central Nervous System
CT Computed Tomography
CV Conduction Velocity
DRCMR Danish Research Centre for Magnetic Resonance
ECS Extracellular Space
EM Electron Microscopy
LM Light Microscopy
MRI Magnetic Resonance Imaging
NIfTI Neuroimaging Informatics Technology Initiative
RAS+ Right-Anterior-Superior positive
ROI Region of interest
SNR Signal-to-Noise Ratio
WM White Matter
XNH X-ray Nano-Holotomography

Contents

Chapter 1

Introduction

1.1 Context

The brain regulates cognitive and physiological functions by constantly processing and communicating massive amounts of information. It is an intricate network of neural circuits wherein neurons transmit information and whose exact configuration and function remains elusive. Neurons are the basic fundamental units of the brain. They are cells consisting of a soma (cell body), dendrites which function to receive signals from other cells, and an axon that connects different brain regions and relays signals between them to other neurons.

The brain and the spinal cord constitute the central nervous system (CNS), which is the body's processing centre. The brain is composed of grey matter and white matter (WM). Grey matter is present in the cerebral cortex, the cerebellar cortex, and other regions of the cerebrum, cerebellum, and brainstem. It contains neuronal cell bodies and dendrites, glial cells, and blood vessels, it synapses on the microstructural level, and its role is to process signals in the brain network. On the other hand, WM is the cabling of the brain. It is comprised of axon tracts that connect the gray matter regions and facilitates signal communication in the brain network. Like grey matter, it also contains blood vessels, astrocytes, oligodendrocytes, and other glial cells. Most of the axons in the WM are 'myelinated', i.e., they are wrapped in layers of lipid, myelin, by an oligodendrocyte which insulates them from the extra-axonal environment. Myelination occurs in segments, with each myelinated section called an 'internode' and separated from other internodes by shorter portions of unmyelinated axon called 'Nodes of Ranvier'. Myelin is a defining feature of the WM and plays an essential role in facilitating rapid and efficient communication between different regions of the brain. Individual internodes exhibit a distribution of g-ratios, the ratio of the outer fiber diameter, including the myelin sheath, to the inner axon diameter. Along the internodes, the myelin thickness is stable, implying that g-ratio variations along single axons are driven by changes in axon diameter.

Different brain regions' structure is adaptable and related to function and experiences, such as aging, learning of visuomotor tasks, training of working memory, and neurodegenerative diseases including Multiple Sclerosis and Alzheimer's disease. The relationship between structure and function is present even at the single-axon level. Axon morphology, specifically diameter and myelin sheath thickness, determines the conduction velocity (CV) of signal propagation

within the brain network. Axons, the communication cables of the brain, are long, tubular structures, often modelled as straight cylinders that constitute a large portion of the brain's WM. Despite being tube-like, light microscopy (LM) and electron microscopy (EM) studies reveal non-constant diameters, shapes, and trajectories in axonal projections over long distances. Axon diameter decreases were observed near Nodes of Ranvier and in response to vacuoles placing tension on the cytoskeleton. The axonal cytoskeleton regulates the diameter downwards or upwards, when exposed to a decrease or increase in tension, respectively. Axonal diameter - and thus CV - varies based on the origin and target of the axon. Furthermore, simulations of diffusion within the intra-axonal space provided evidence that axon diameter estimates based on the diffusion magnetic resonance imaging (MRI) signals perpendicular to the axons, including those in the corpus callosum, were subject to a time-dependent overestimation. Certain neurodegenerative diseases target specific axon sizes. Small-diameter axon degeneration is observed in Multiple Sclerosis, while Amyotrophic Lateral Sclerosis affects the thickest ones. Therefore, axon diameter can serve as a valuable biomarker for diagnosing these conditions. Cell clusters, other axons, and blood vessels cause significant trajectory changes in the axons. Disease or trauma-induced neuroinflammation leads to an accumulation of cells in the extra-cellular space (ECS), which increases the density of extra-axonal structures that can alter axonal trajectories and bias axon diameter measurements. The diameter and trajectory of axons exhibit variation along their length, often attributed to local microstructural obstacles.

Neuronal communication occurs through signal transmission and action potential (AP) generation. Sensory cues and, most commonly, presynaptic neurons can trigger APs. Thicker myelin sheaths in nerves with smaller diameters conducted APs faster than thinner myelin sheaths in nerves with larger diameters, demonstrating the dependence of CV on myelin thickness in addition to axon diameter. Myelin enhances the CV by more than a factor of 10 in fibers of more than a few microns in diameter, compared to unmyelinated axons of the same size. Myelin increases CV by enhancing signal conduction, while internodal distance, axon diameter, Node of Ranvier length, and g-ratio also affects CV. Axonal structural properties are closely linked to CV and each other. For example, in a larger diameter axon, the flow of charge along it experiences less resistance and can depolarize more distal membrane segments, meaning that the internodal distance should also increase to optimize the signal conduction.

At present, diffusion MRI is the sole method to measure axon diameter in the living brain. It utilizes water molecule diffusion to investigate white matter microstructure. Nonetheless, since axon diameters in the brain are on a micrometer scale, whereas MRI voxels for in vivo studies are on a millimeter scale, biophysical models describing the expected geometries of the WM compartments must be fitted to the acquired diffusion MRI signal to extract microscale information. Additionally, research has shown that diffusion MRI-based axon diameter estimates tend to overestimate those derived from LM and EM; possibly due to a mismatch between biophysical model design, questionable validity of comparing metrics from disparate modalities and experimental conditions, and a possible impact of MRI hardware limitations on axon diameter measurements. In EM, axons are only tracked for up to 20 µm, a fraction of their length in MRI voxels.

A 3D validation of the WM anatomy and the biophysical models mentioned is necessary to test the validity of modeling axons as cylinders and of axonal structure-function relations.

1.2 Objectives

The primary aim of this dissertation is to explore the application and effectiveness of 3D convolutional neural networks for 3D axon segmentation in XNH volumes from the splenium of the vervet monkey brain. To accomplish this, the specific objectives of this report are: to study, preprocess and effectively handle the available sample image volumes, to develop a training and evaluation framework for assessing the segmentation performance of the 3D U-Net model on the available data under various scenarios, and to obtain an overview of the morphological characteristics of the predicted axons. Ultimately, the final objective of this research is to achieve a dense and robust annotation of axonal structures within the given volumes.

1.3 Organization of the dissertation

The coming dissertation begins with a thorough examination of the available XNH synchrotron data, which was previously analyzed by M. Andersson et al. in [2]. This dataset and the preprocessing steps are explained in detail in chapter 2, along with a comprehensive explanation of the employed methods, including the 3D U-Net model and the concept of 'cross-hair' filters from [13]. The experimental setup and training environment, encompassing patch-based data loading, performance metrics, and data augmentation, are included in this same chapter. The outcomes of various experiments are gathered in chapter 3, which investigates different scenarios, including non-augmented and augmented baseline U-Net models, as well as the integration of 'cross-hair' filters. This chapter also examines the morphology of the predicted axons. The final two chapters, 4 and 5, are dedicated to providing a comprehensive comparison and overview of the outcomes from the previous chapter and developing the conclusions from the research by summarizing the dissertation as a whole, respectively.

Images that provide visual support and evidence for the visualizations and statements made throughout the dissertation are collected in Appendix A and B.

Chapter 2

Methods

In this study, PyTorch [10] has been selected as the open-source framework of choice due to its improved usability in comparison to TensorFlow [1]. TorchIO [11], a unified Python library specialized in data loading, preprocessing, augmentation, and patch-based sampling of medical images in the context of deep learning, has been utilized for efficient data handling. This library focuses on the ease of use for researchers by providing a user-friendly framework that simplifies the process of working with medical imaging data in PyTorch. Its modularity and flexibility allowed to easily customize existing methods to make them usable in the context of this thesis. Additionally, TorchIO considers medical images as physical objects in the space taking into account spatial metadata. The provided data is stored in Neuroimaging Informatics Technology Initiative (NIfTI) file format.

In this chapter, the methods used for the study are presented. Firstly, the data utilized is described, including the XNH imaging technique, and the data preprocessing steps implemented to prepare the data for further analysis are outlined. Subsequently, the different models that will be explored are presented. Following that, details on the training process are provided, including the data loading framework and the patch-based training, as well as loss functions, performance metrics, and optimizer. Finally, the chapter concludes by presenting the experimental setup and experiment tracking platform.

2.1 Data

Volumetric data is abundant in biomedical data analysis. Annotation of such data with segmentation labels is challenging since only 2D slices can be handled at a time. Slice-by-slice annotation is inefficient and tedious due to the redundant information in neighboring slices. Thus, complete 3D volume annotation is not an effective method to generate extensive and generalized training data.

In many biomedical applications, a few images can productively train a network that generalizes reasonably well on account of the presence of repetitive structures with corresponding variations in each image. Weighted loss functions and data augmentation facilitate training networks with sparsely annotated data.

2.1.1 Synchrotron X-ray Nano-Holotomography XNH

The study by M. Andersson et al. [3] showcases the interplay between extra-axonal structures and the micromorphology of axons through the segmentation and analysis of high-resolution 3D volumes obtained from the WM of a vervet monkey brain with synchrotron X-ray nano-holotomography (XNH), and utilizes the findings to validate diffusion MRI techniques for estimating axon diameters. The study made both anatomical and technical contributions. XNH technique enables the acquisition of high-resolution 3D maps in the holographic imaging regime of the inner structure of nano-resolution scale samples, utilizing X-rays. The tissue samples were meticulously prepared before the synchrotron experiments to minimize image artifacts and X-ray absorption while preserving structural features and maintaining contrast to the present biological structures.

However, the constraints of the study conducted by M. Andersson et al. indicate that additional research is needed. First, the combination of insufficient signal-to-noise ratio (SNR) and limited image resolution challenged the robust segmentation of axons smaller than 2 µm, requiring further investigation of the morphological characteristics of smaller axons, considered to account for most of the ADD in WM. Second, the time-consuming process caused by the image resolution and size handling resulted in the segmentation of only 54 axons, despite the XNH volumes containing a much larger number. Although the segmentation was semi-automatic, it had to be corrected manually in Nodes of Ranvier and regions of lower contrast where the myelin boundary between the axoplasm and extracellular space (ECS) was unclear.

There is also a potential to gain more from the already analyzed XNH volumes by the implementation of a Convolutional Neural Network (CNN) segmentation method on the XNH volumes as mentioned in [3], resulting in the segmentation of thousands of axons. This observation will be studied in detail in this project.

2.1.2 Dataset

The dataset used for the present thesis is the 'Axon Morphology Dataset', which contains XNH volumes of the splenium of the vervet monkey brain. This dataset was retrieved from the Danish Research Centre for Magnetic Resonance (DRCMR) and previously utilized by M. Andersson et al. in [3] and [2].

To obtain the dataset, the DRCMR mapped the WM microstructure using synchrotron XNH and acquired several image volumes from different WM regions. No image registration was required as the sample remained stationary during sampling. The main focus is on the XNH volume of the splenium region with an isotropic voxel size of 75 nm.

A preliminary segmentation of 54 axons was performed in the XNH volume of the splenium region, with a cylindrical field of view of height and diameter of 153.6 µm. Segmentation of cells, blood vessels, and vacuoles was possible within a volume of $153.6 \times 153.6 \times 584.5$ µm^3 by combining four adjacent and overlapping XNH volumes. Six large axons (>2 µm diameter) longer than 660 µm were tracked and analysed within the extended volume, providing information on the long-range axon behavior.

Prior to the axon segmentation, the XNH volumes underwent downsampling by a factor of 5 via extraction of every fifth slice and performing a slicewise cubic interpolation and Gaussian smoothing with a kernel width of 5 pixels. As a result of this downsampling process, the dimensions of each of the four individual volumes became $410 \times 410 \times 410$ voxels, with an isotropic voxel size of 375 nm, enabling the entire volumes to be loaded into image processing software. The dimensions of the combined volume resulted in $410 \times 410 \times 1532$ voxels.

Next, a layered surface segmentation method was used to extract the axons and the extra-axonal structures in the study by M. Andersson et al. [2]. The rough segmentation of the axons from the volumes was performed using the adaptive paintbrush tool in ITK-Snap [14], while cell clusters, blood vessels, and vacuoles were extracted by an intensity- and morphology-based approach in MATLAB. All axons exceeded 120 µm in length, and ADs were quantified every 150 nm. The dissertation will primarily focus on the fourth fraction of the combined XNH volume shown in Figure 2.1a, which will be referred to as 'the XNH volume' henceforth. This volume section provides both axon and extra-axonal structure segmentations as ground truth for analysis. The orientation of the volumes corresponds to the Right-Anterior-Superior positive (RAS+) system, widely used in various imaging modalities including MRI and Computed Tomography (CT). The dataset presented herein will serve as the study sample for this thesis.

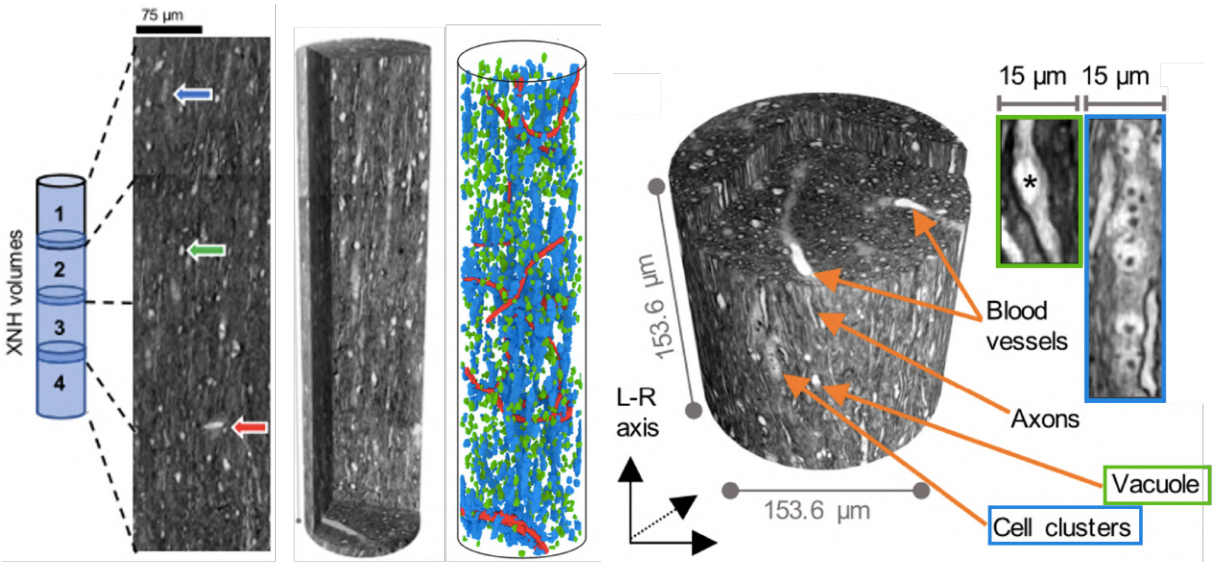
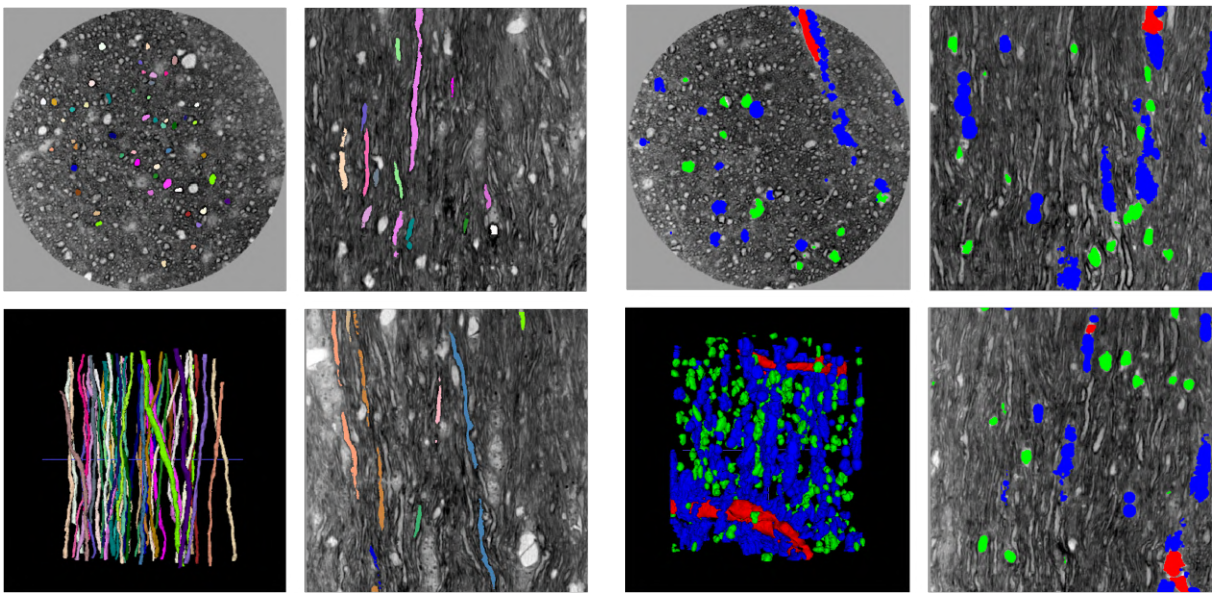


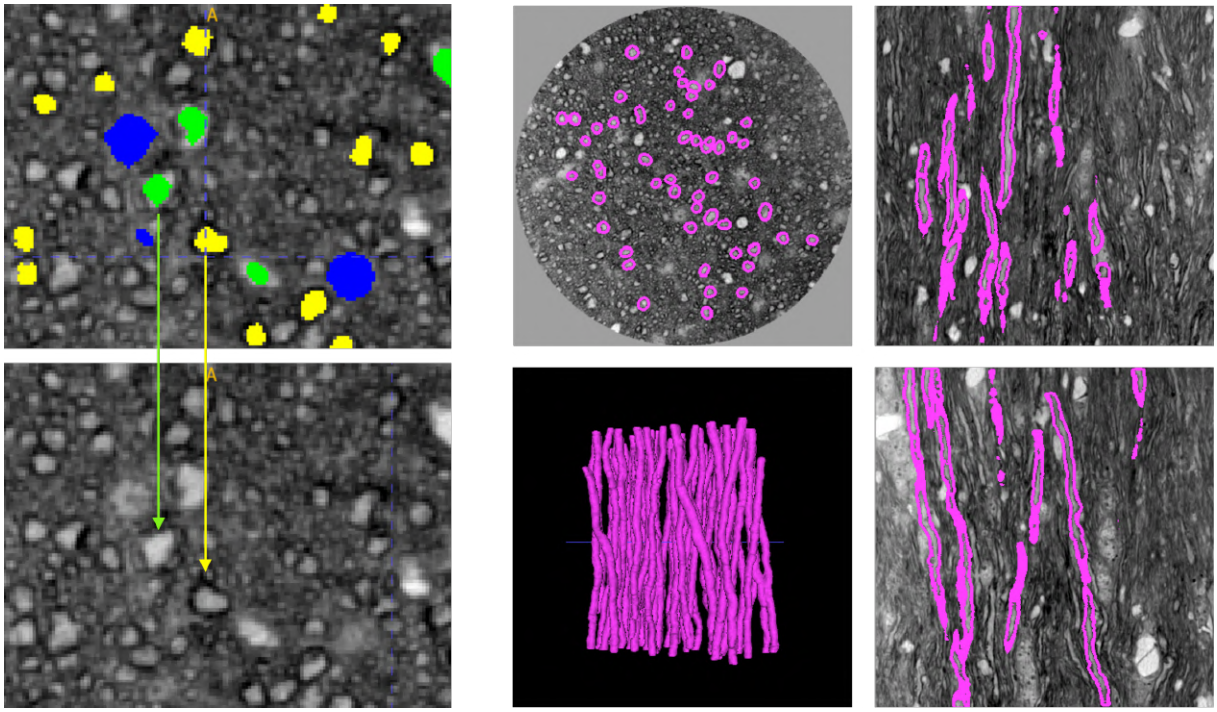
Figure 2.1: (a) Combined volume composed of four overlapping XNH volumes (13.5 µm overlap), and the segmentation of extra-axonal structures: blood vessels (red), vacuoles (green), and cell clusters (blue). (b) Fourth fraction of the combined XNH volume providing a closer view of vacuoles and cell clusters. Figures adapted from [3] and [2].

The XNH volume showed distinguishable myelinated axons, cell nuclei, vacuoles, and blood vessels, depicted in Figure 2.2. The segmentation revealed an even spatial distribution of vacuoles (1.04% volume fraction), whereas cells (3.21% volume fraction) were clustered, particularly along the axonal direction, with many closely associated or 'anchored' in the blood vessels (0.53% volume fraction). The study will exclude unmyelinated axons on account of their relatively small dimensions.



(a) (b)

Figure 2.2: (a) Segmentation of 54 axons in the volume fraction from (Figure 2.1b) visualized using the ITK-Snap software application. (b) Segmentation of extra-axonal structures in the volume fraction from (b) visualized using ITK-Snap.



(a) (b)

Figure 2.3: (a) Traverse sections of axons (yellow) and vacuoles (green) exhibit similarities in their internal structure. (b) Segmentation of the myelin sheath surrounding the 54 axons segmented in Figure 2.2a.

Vacuoles present ellipsoidal shape with a mean diameter of 7.8 µm, while the mean axon diameter ranges between 2.1 and 3.8 µm in the primitively segmented axons [2]. Axons exhibit structural similarities to vacuoles in transverse sections due to their circular shape and light interior (see Figure 2.3a). Although their diameter can serve as a reliable differentiating factor, the distinction between the two can be enhanced by considering the unique myelin sheath surrounding axons. A fifth label has been added to the segmentation for the myelin to assist axon segmentation (see Figure 2.3b). While the myelin sheath remains constant within each internode but is not uniform across different internodes, it has been depicted as consistent (constant thickness) everywhere across all axons in this research. The aforementioned could be a potential limitation of the study, and refining the myelin segmentation via the exclusion of the masked highest voxel values from the original image may impact model performance.

Blood vessels differ significantly from axons in shape, being considerably larger. Similarly, cell clusters are generally larger than axons and can be easily distinguished by the presence of dark dots (cell nuclei) distributed within their shape (Figure 2.1b).

2.1.3 Data preprocessing

This study utilizes three raw data files: one for the downsampled XNH volume, another for the 54 axon segmentation, and the final one for the extra-axonal structure segmentation. While the first two files contain information corresponding to the fourth fraction of the combined XNH volume presented in subsection 2.1.1, the last one was only available for the entire combined volume and had to be cropped to this specific section. As a result, each file contained an image with dimensions of $410 \times 410 \times 410$ voxels, with 0.38 µm spacing corresponding to the isotropic voxel size of 375 nm.

Preprocessing the data is essential in machine learning to ensure data quality and consistency, which helps improve model performance, reduce complexity, and create a more reliable and effective foundation for training and generalizing machine learning models.

Starting with the XNH 3D volume, the image's intensity values range from -105 to 122, with a mean value of -20.28 and a standard deviation of 15.29. It is important to note that the sampled 3D volume in this study has a cylindrical shape. This means that the information is contained in a cylindrical volume, while the complementary space in the cubic $410 \times 410 \times 410$ voxel image volume contains just background voxels (with intensity value 0). To improve the handling of foreground voxels and exclude empty voxels during preprocessing, a cylindrical binary mask was created to fit the image volume. Initially, the zero-intensity voxels were removed from the whole cubic volume. Subsequently, a binary closing operation was applied to seal the zero values within the sampled volume, resulting in a compact mask. The mean voxel intensity within the masked volume became -25.79 with standard deviation of 12.47. The resulting cylindrical binary mask, referred to as the foreground mask hereafter, along with the intensity histograms of the initial volume before and after masking, can be visualized in Figure 2.4.

The next step was to normalize the masked cylindrical XNH volume by subtracting its mean and dividing by its standard deviation to then rescale the voxel intensities to positive values within the range of 0 to 255. The mean voxel intensity within the cylinder became 88.98, with a standard deviation of 14.00. Background voxels outside the cylindrical volume remained at a value of zero.

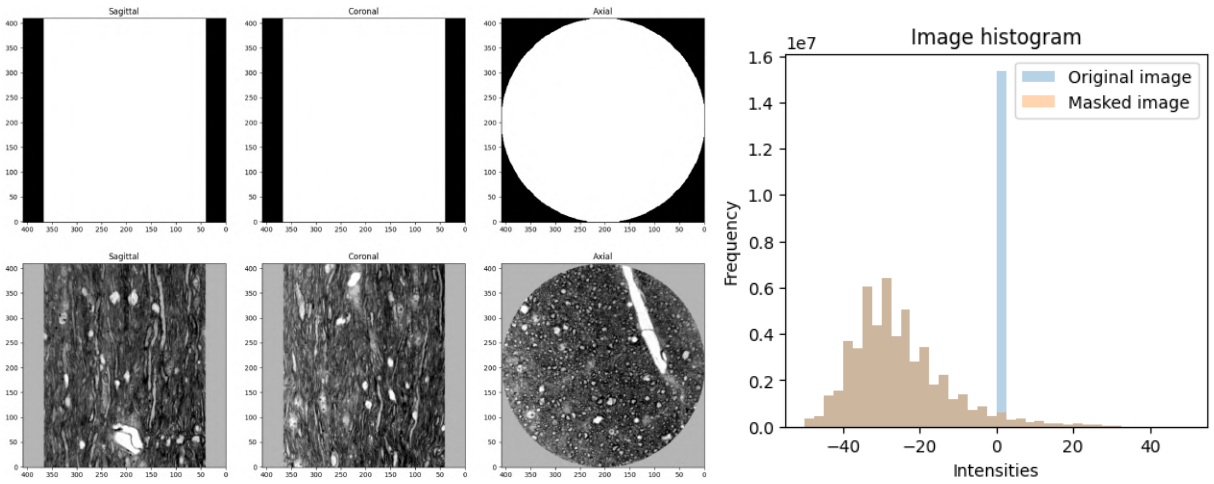


Figure 2.4: (a) View of the cylindrical mask and the original image in their 80$^{\text{th}}$ sagittal, coronal, and axial slices. (b) Histogram of voxel intensities in the cylindrical region of interest (ROI) masking and original volumes, ranging from -50 to 50 intensity values (excluding intensity values in the extremes).

The axon segmentation volume assigns label value 0 to unlabeled voxels and 1 to 54 for each individually segmented axon, as illustrated in Figure 2.5b. A binary label map was generated to represent the entire axon class (see Figure 2.5c). The myelin was extracted from the axon segmentation by dilating the axon mask and subtracting the non-dilated axon mask from it. This step is expected to assist the model in accurately segmenting axons based on this distinctive characteristic, Figure 2.5d. As mentioned in subsection 2.1.2, in this study, the myelin was consistently segmented along and across all axons.

Despite the focus of the project is to get the axon segmentation to study their 3D morphology, a multiclass segmentation approach that includes blood vessels, vacuoles, cell clusters, axons, and myelin, can be more effective in distinguishing axons from other extra-axonal structures. The segmentations of axons, myelin, and extra-axonal structures were combined into a single label map with labels ranging from 0 to 5, representing unlabeled (0), blood vessel (1), vacuole (2), cell cluster (3), axon (4), and myelin (5). During label map merging, a total of 3062 voxels (<0.005%) exhibited overlapping axon and extra-axonal labels. Upon visual inspection of the labeled volume, it was decided to keep the axon label, due to the observed overlap being attributed to relatively imprecise labeling of the extra-axonal structures. The merged map and the mask overlap are depicted in Figure 2.6.

As the sole sample volume for this study, the processed images and segmentations of the sample were stored locally in NIfTI format, the same format as the initially provided data. They were subsequently loaded from memory in their modified state for each experimental run to optimize computation time. The image volume was converted to 32-bit floating-point numbers (float32), which is the lowest precision data type supported by the PyTorch convolutions used later, and the segmentations to 8-bit unsigned integer numbers (uint8) to optimize memory usage.

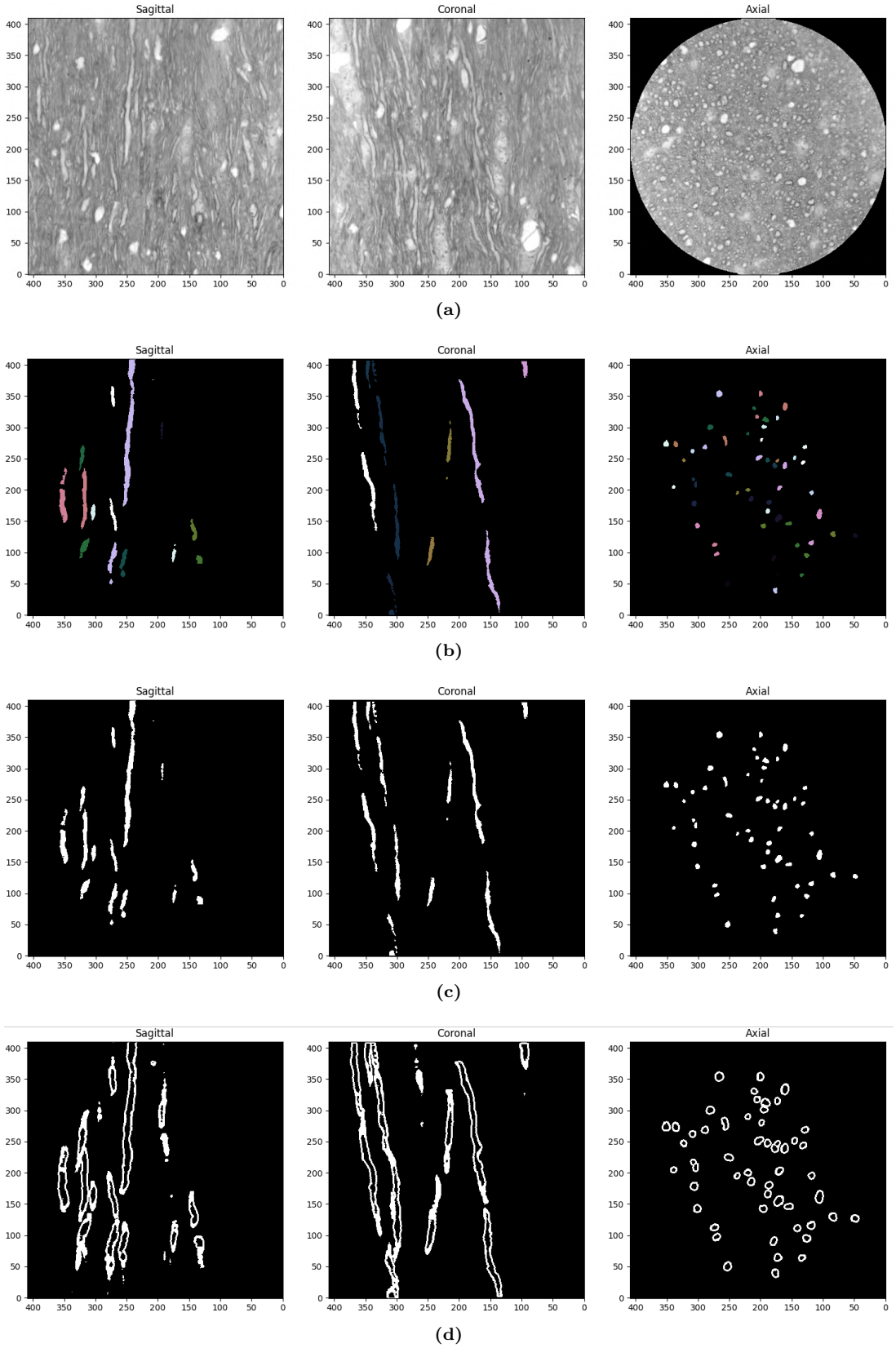


Figure 2.5: (a) Normalized and rescaled XNH volume. (b) Provided axon segmentation. (c) Generated binary axon segmentation. (d) Generated myelin segmentation.

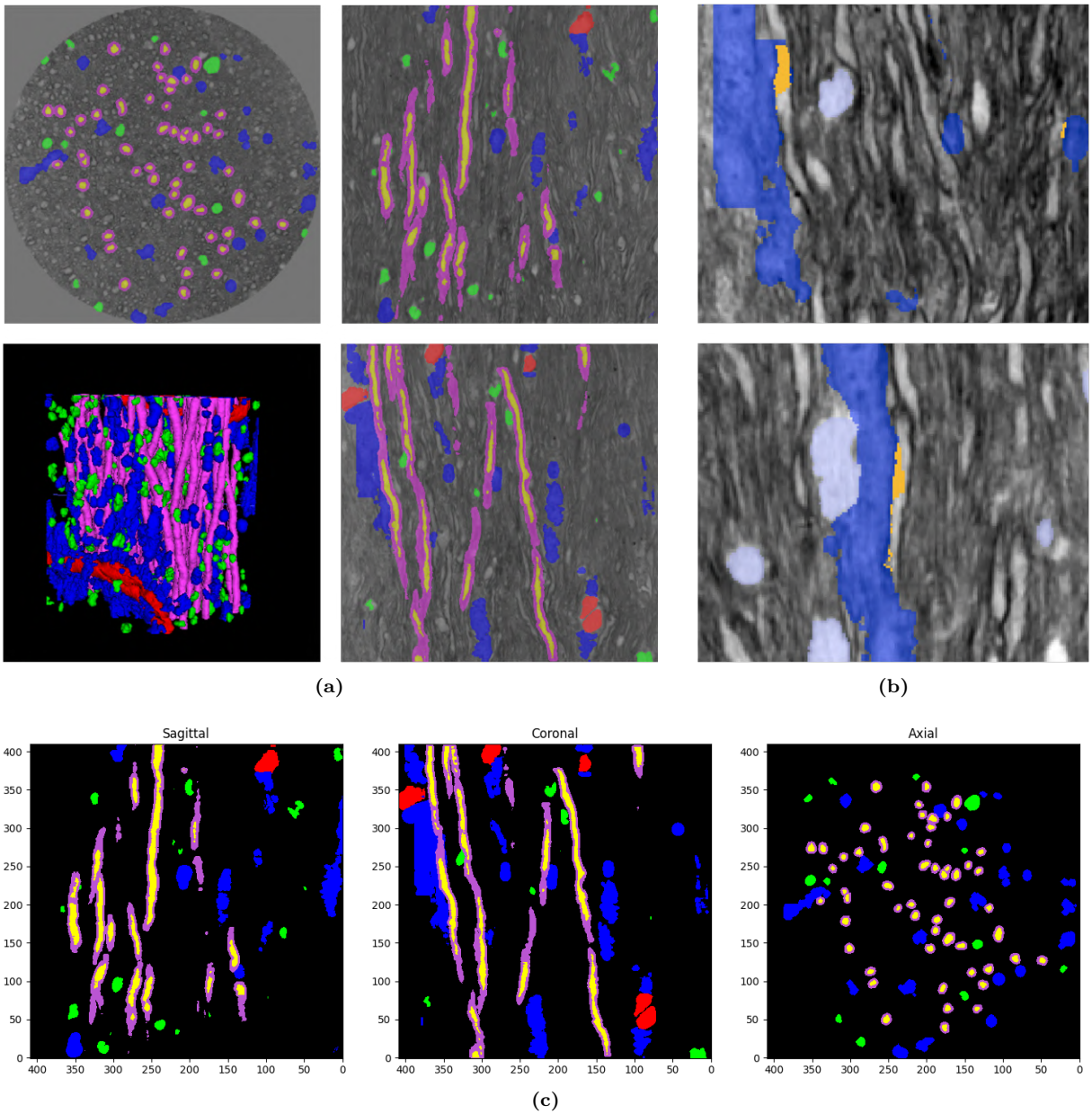


Figure 2.6: (a) Segmentation of, blood vessels (red), vacuoles (green), cell clusters (blue), axons (yellow), and myelin (magenta) visualized in ITK-Snap. (b) Axonal and extra-axonal mask overlap. (c) Merged label mask containing all axonal and extra-axonal structures.

2.1.4 Subject

The TorchIO library introduces the `Subject` data structure, which serves to store subject-associated images and metadata. This class allows storing the image volume, multiclass segmentation, and foreground mask, as well as shape, spacing, orientation, and memory information. If necessary, additional subject details such as age, name, and ID could also be stored. Additionally, it offers high usability through its transform reproducibility and invertibility, in addition to the availability of plotting and image-loading methods. The images stored within a subject can belong to the `ScalarImage` or `LabelMap` image subclasses, representing voxel values as scalars and categorical labels, respectively. Scalar images are ideal to store the image volume, while

the label maps contain the segmentation masks. Transforms are applied to every image within the subject, nevertheless, intensity transforms are not applied to label images. Henceforth, the concept of subject will refer to an element containing the processed image volume, the multiclass segmentation, the foreground mask, and related metadata.

2.2 Models

The quantification of 3D axonal morphology provides valuable insights into anatomical trends previously observed in 2D axon diameter investigations and emphasizes the importance of considering the third dimension for accurately describing the structure and function of individual axons. Axon segmentation requires accurate identification and separation of axons from surrounding tissue in 3D images.

For axon segmentation, CNNs are a popular approach in deep learning. In this thesis, 3D CNNs were selected for study over 2D CNNs due to various reasons. First, a 3D CNN surpasses a 2D CNN in capturing the spatial context and shape of axons, as it considers the whole image volume instead of processing each slice independently. This is crucial for axon segmentation, where the shape and orientation of axons can vary significantly in the 3D space. Second, a 3D CNN can improve segmentation accuracy by utilizing correlated information for neighboring slices in a 3D image. Last, a 3D CNN can better distinguish between axons and surrounding tissue that may have similar texture or intensity in a single 2D slice, thus, a 3D CNN can potentially decrease false positives/negatives in axon segmentation more effectively compared to a 2D CNN.

However, 3D CNNs have a downside as they are computationally expensive and require more training data compared to 2D CNNs.

2.2.1 3D U-Net

The proposed network, based on the U-Net architecture introduced by Ö. Çiçek et al. [5], extends the original U-Net architecture proposed by O. Ronneberger et al. [12] by replacing all 2D operations with their 3D counterparts. This extension is motivated by the fact that medical images, such as the sampled XNH volume, possess an inherent 3D structure for which slice-wise processing would be suboptimal. The objective of the introduced network is that it learns from the existing sparse annotations and provides a dense 3D segmentation.

The U-Net is a deep convolutional network architecture designed for fast and precise end-to-end image segmentation. It was initially designed for the segmentation of biomedical images and it surpassed prior approaches such as the sliding-window CNN. Figure 2.7 illustrates the network architecture implemented in this project.

The network processes the raw input image volume to produce the segmentation map. Most operations are $3\times3\times3$ (or $5\times5\times5$) 3D convolutions followed by a batch normalization operation and a rectified linear unit (ReLU) non-linear activation function. These convolution blocks double the number of feature channels. Subsequently, the Max-Pooling operation reduces the dimensions of the feature map by propagating the maximum activation from each $2\times2\times2$ window

to the next feature map. The sequential application of convolutions and max-pooling leads to spatial contraction where the network learns to focus on the underlying features while reducing the influence of their spatial location. This section of the network is commonly referred to as the 'encoder', and is responsible for gradually compressing information into a lower-dimensional representation and processing it at higher levels of abstraction.

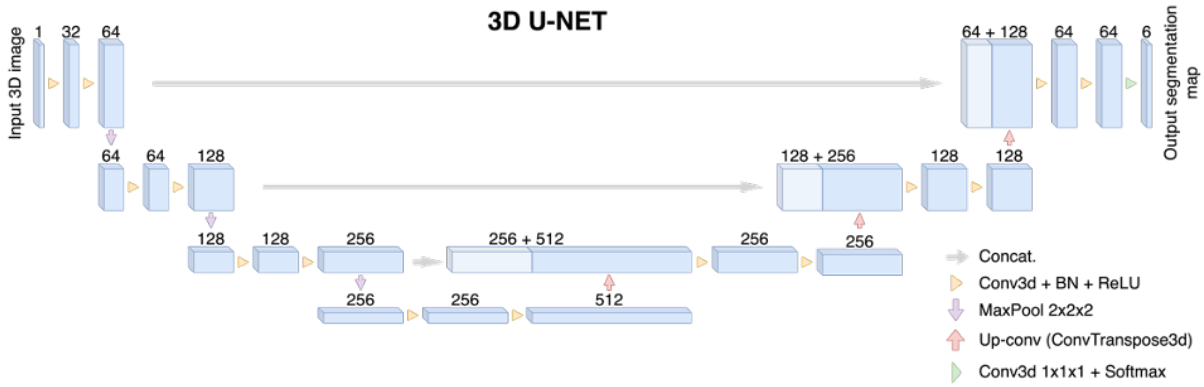


Figure 2.7: Each blue box corresponds to a multi-channel feature map, with the number of channels indicated on top. Lighter blue boxes represent copied feature masks obtained from skip connections, which are depicted by grey arrows. Different arrows symbolize the operations within the network.

While standard classification networks typically terminate here by mapping all features into a single output vector, the U-Net architecture includes an additional expansion path, also named 'decoder', specifically designed for creating a high-resolution segmentation map. The expansion path incorporates a combination of up-convolutions and concatenation with high-resolution features from the contracting path. These connections, known as skip connections, facilitate information flow from the encoder to the decoder, thereby improving model predictions by combining both low-level and high-level information. As an example, texture differences and edge information aid in object boundary detection, while high-level knowledge about an object's class helps in precisely segmenting voxels according to their corresponding object class. The up-convolution, a transposed 3D convolution, applies a learned kernel to each feature vector and maps it to a $2 \times 2 \times 2$ output window. The resulting mapping is then concatenated with the equal resolution layer's features from the contracting path, and passed through a 3D convolution block including batch normalization operations and ReLU activation maps. On the last layer of the network, a $1 \times 1 \times 1$ convolution reduces the output channels to six, one for each class to segment: background, blood vessel, vacuole, cell cluster, axon, and myelin. Then, the Softmax activation function rescales the raw model outputs (logits) to a probability distribution over the classes, with each class having a probability between 0 and 1 and the sum of probabilities across all classes equaling 1.

Convolution padding ensures that the output size remains the same as the input size. This U-Net implementation will allow the classification of each voxel in the input image to the label it represents, i.e., voxel-level classification. Note that batch normalization (BN) is applied before each ReLU activation function to expedite convergence; it helps maintain a stable input distribution to the activation function and aid in training the network more effectively. During training, each batch is normalized using its own mean and standard deviation, and the global statistics are updated accordingly.

Compared to 2D convolutions, the additional dimension in 3D convolutions significantly increases the computational complexity, memory requirements, and training time, by requiring the processing of a drastically larger number of parameters in each convolutional operation. The Table 2.1 shows that a 3D patch with dimensions $128 \times 128 \times 128$ voxels requires 133.33 times more memory (8 MB) compared to a 2D patch of dimensions 128×128 (0.06 MB). Furthermore, the number of trainable parameters increases drastically from ∼8.5 million to ∼25.6 million from the 2D case to the 3D case for a $3 \times 3 \times 3$ kernel, and from ∼23.7 million to ∼118.4 million for a $5 \times 5 \times 5$ kernel, respectively. Consequently, the estimated total size requirements, which represents the total memory footprint required to store the entire neural network including input, intermediate and output tensors, and the parameters, increases in between 6 300% and 8 000% for kernels of size 3 and 5. The concept of 'cross-hair' filter is introduced in the next section with the objective of reducing the memory requirements and computational costs of 3D CNNs while maintaining their benefits over 2D CNNs.

Model	Trainable Parameters		Input size (MB)	Estimated Total Size (MB)	
	Kernel Size			Kernel Size	
	3	5		3	5
2D U-Net	8 539 360	23 711 456	0.06	217.14	275.01
3D U-net	25 607 968	118 537 056	8.0	17 337.69	17 692.18

Table 2.1: Number of trainable parameters, input size, and estimated total size of the U-Net network architecture presented in Figure 2.7, with input volume shapes of 128×128 and $128 \times 128 \times 128$ voxels for the 2D and 3D networks, respectively. Computed by `torchsummary`.

A popular idea would be to perform slice-wise 2D CNNs; however, this approach discards valuable 3D context information crucial for tracking tubular structures in 3D.

2.2.2 Cross-hair filters

G. Tetteh et al. presented DeepVesselNet [13], an architecture with the objective of overcoming the execution speed and high memory challenges, faced when extracting vessel trees and networks, and corresponding features, in 3D angiographic volumes using deep learning. To outperform full 3D networks in terms of computational cost, the idea of 2D orthogonal cross-hair filters is employed, which utilize 3D context information at a reduced computational load.

As presented in Figure 2.8, the idea of cross-hair filters involves the utilization of three intersecting 2D planes centered in each target voxel. This concept helps to mitigate memory and speed problems of classical full 3D networks while learning from the information across slices in volumetric data and maintaining accuracy. While the slice-wise existing ideas would extract 2D planes in the three directions at a preprocessing step and feed them into the network, the cross-hair filters are implemented on a layer level within the network, helping to retain the

three-dimensional information through the network.

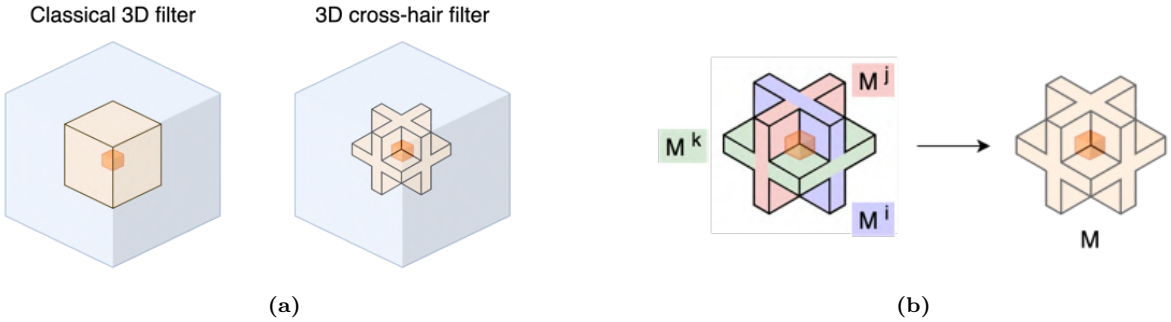


Figure 2.8: (a) Graphical representation of a classical 3D convolution filter (on the left) and a 3D convolution with cross-hair filters (on the right). (b) The cross-hair kernel is composed by three orthogonal 2D filters.

Considering a classical 3D convolutional kernel shape of (k_x, k_y, k_z), the mathematical representation of a 3D convolution operation $*$, as presented in [13], can be expressed as follows:

$$I * M = A = \{a_{ijk}\} \tag{2.1}$$

where I is the input image, M is the kernel or filter, A is the convoluted image, and $\{a_{ijk}\}$ is the element with index (i, j, k) in matrix A. Then, a classical 3D convoluted element can be expressed as:

$$\begin{aligned} a_{ijk} &= \sum_{r=1}^{k_x}\sum_{s=1}^{k_y}\sum_{t=1}^{k_z} I_{(R,S,T)} M_{(r,s,t)} \\ R &= i + r - \left(1 + \left\lfloor \frac{k_x}{2} \right\rfloor\right) \\ S &= j + s - \left(1 + \left\lfloor \frac{k_y}{2} \right\rfloor\right) \\ T &= k + t - \left(1 + \left\lfloor \frac{k_z}{2} \right\rfloor\right) \end{aligned} \tag{2.2}$$

where $I_{(R,S,T)}$ is the intensity value of image I at voxel position (R, S, T), and $M_{(r,s,t)}$ is the value of the kernel M at position (r, s, t). Conversely, the elements of a convolution operation $*$ with a cross-hair filter that approximates the standard 3D convolution can be expressed by:

$$\begin{aligned} a_{ijk} = &\sum_{s=1}^{k_y}\sum_{t=1}^{k_z} I_{(i,S,T)} M^i_{(s,t)} + \sum_{r=1}^{k_x}\sum_{t=1}^{k_z} I_{(R,j,,T)} M^j_{(r,t)} \\ &+ \sum_{r=1}^{k_x}\sum_{s=1}^{k_y} I_{(R,S,k)} M^k_{(r,s)} \end{aligned} \tag{2.3}$$

where M^i, M^j, M^k are the 2D convolutional cross-hair kernels that will intersect to provide an approximation of the classical 3D kernel M along the i-th, j-th, k-th axes, respectively.

The standard 3D convolution requires $k_x k_y k_z$ multiplications and $k_x k_y k_z - 1$ additions for each voxel. In contrast, the cross-hair 3D convolution involves in $k_y k_z + k_x k_z + k_x k_y$ multiplications and $k_y k_z + k_x k_z + k_x k_y - 1$ additions; two of the additions result from adding the output of the

three 2D filters (M^i, M^j, M^k) together. Applying Equation 2.3 independently for each voxel is inefficient due to redundant use of memory. For a more efficient implementation, each of the three intersecting 2D filters is applied to the corresponding sagittal, coronal, and axial planes of the input volume; in this manner, only one volume is loaded in memory while kernels are rotated to match the slices in different orientations. Figure 2.9 provides a visual representation of cross-hair filters and their implementation. In practice, three 3D convolutions are added with kernel dimensions $(1, k_y, k_z)$, $(k_x, 1, k_z)$, and $(k_x, k_y, 1)$, respectively, for each convolution operation in the U-Net network architecture (see Figure 2.8b).

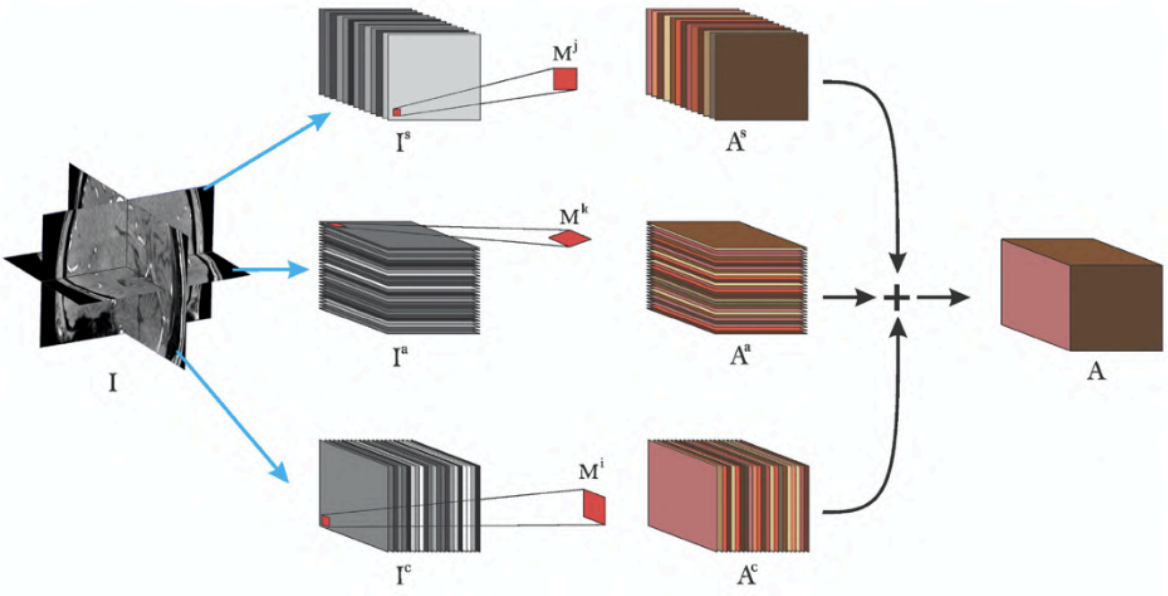


Figure 2.9: Graphical representation of the implementation of a 3D cross-hair filter. Grayscaled stacks refer to the input to the layer, and brown colored slices refer to extracted features after convolution operation. Figure retrieved from [13].

Now, let k_{m1}, k_{m2}, k_{m3} be the sizes of the kernel M such that $k_{m1} \geq k_{m2} \geq k_{m3}$, then:

$$k_y k_z + k_x k_z + k_x k_y \leq 3(k_{m1} k_{m2}) \leq k_x k_y k_z \tag{2.4}$$

where strict inequality holds for $k_{m3} > 3$. A more detailed description of the previous equations can be found in [13]. While for a filter $(k_x, k_y, k_z) = (3, 3, 3)$ there is no significant improvement, for larger filters there is a considerable decrease on the number of operations as it can be see in a simple example in Table 2.2.

Convolution	No. of operations			
	Kernel Size			
	3		5	
	sums	multiplications	sums	multiplications
Classical 3D	26	27	124	125
Cross-Hair 3D	26	27	74	75

Table 2.2: Number of operations per voxel of the resulting image involved in classical 3D convolution filters versus cross-hair convolution filters.

Taking inputs from Table 2.1, we compare in Table 2.3 number of parameters reduction when choosing cross-hair convolution filters over classical 3D convolution filters. Focusing on a kernel size of 5 voxels, the number and size of trainable parameters gets reduced on approximately by 40%. The memory requirements will be analysed in chapter 3.

Model	Trainable Parameters	
	Kernel Size	
	3	5
3D U-Net	25,607,968	118,537,056
Cross Hair 3D U-net	25,607,968	71,124,256

Table 2.3: Number of trainable parameters in the U-Net network architecture presented in Figure 2.7 with an input volume shape of $128 \times 128 \times 128$, utilizing cross-filter convolutions and without their usage.

The cross-hair filters described in this subsection were implemented in PyTorch, inspired by the approach used in DeepVesselNet [13] for TensorFlow.

2.3 Training

2.3.1 Patch-based training

Using the entire image volume of size $410 \times 410 \times 410$ voxels as input to a 3D CNN is not feasible due to limitations in memory and computational resources. To address the memory constraints associated with the large volume size and ensure more efficient processing, it is necessary to extract smaller patches from the volume. These extracted patches can then be fed to the network for further analysis. By working with patches of an appropriate size, the network can still effectively learn spatial dependencies within the volume and produce accurate predictions.

The extracted patch should be large enough to contain entire or representative shapes of the different elements. The classes that can be more challenging to differentiate are axons and vacuoles due to their similar circular shape and diameter in a transverse cut, as stated in subsection 2.1.2. In the original full resolution volume, segmented axons' average diameters range from 2.1 µm to 3.8 µm and vacuoles' mean diameters from 4.2 µm to 7.8 µm. Thus, with voxel size of 0.38 µm, patches exceeding dimensions of $21 \times 21 \times 21$ voxels can capture the confusable transversal shape of these structures. Nevertheless, it is important to capture the characteristic longitudinal shape of axons. Considering a consistent spatial distribution of labeled axons and extra-axonal structures within the volume and contemplating their shapes and sizes, a patch of size $64 \times 64 \times 64$ voxels will contain the necessary information of study. However, to assess and compare the models' performance, experiments will initially be conducted using patch sizes of $64 \times 64 \times 64$ and $128 \times 128 \times 128$ voxels.

TorchIO provides different 'sampler' modules that allow to extract patches of specific dimensions across a whole volume. None of the available samplers was completely suitable for the current project; therefore, a customized grid sampler was developed by modifying the source code of the

`torchio.GridSampler` class. The original grid sampler extracts every patch from a grid across the entire volume, but neither it nor any other available random samplers effectively handle the cylindrical shape containing the information in the sample image, resulting in frequent selection of patches containing scarce information. This issue is further extended when rescaling the volumes for data augmentation, as it will be observed later in section 3.2. The desired patch overlap size can be specified.

In order to refer to the patches that contain most information within the image volume, the concept of 'legal' patch will refer to those whose eight corners fall inside the foreground mask. Recall that the original foreground mask is cylindrical but may undergo transformations during data augmentation. The customized grid sampler first identifies the valid patch locations and then randomly selects a patch from the volume at one of those locations. If no legal patches are available in a volume, the sampler will relay into the `WeightedSampler` from TorchIO and randomly extract patches from the volume based on the probability map associated with the foreground mask. In this case, the probability of sampling a patch centered on a particular voxel is determined by the corresponding value in the probability map, ensuring that the center voxel of the extracted patch always contains tissue information.

2.3.2 Data loading framework

TorchIO's `Queue` is a crucial component for training patch-based models for image segmentation. It facilitates stochastic patch-based training by enabling the sampling of multiple patches from a volume, eliminating the need to prepare the volume repeatedly for each patch extraction. The patches are stored in the queue until the next training iteration, eliminating the requirement for local or cloud storage and providing memory and computational advantages. In this queueing system, samplers generate random patches from volumes in the subjects dataset. Furthermore, the queue supports parallel data loading, ensuring efficient utilization of system resources and faster processing times.

In the `Queue` method, at the beginning of each training epoch, a PyTorch loader queries the datasets copied in each process of loading and processing the volumes in parallel on the CPU. Next, the queue list is filled and shuffled with patches extracted by the sampler from the different subjects in the dataset, limited to a specified maximum queue length. A separate data loader is employed to group patches in the queue into batches and feed them into the neural network. When the queue gets emptied as patches are used, it is refilled with new patches.

The maximum length of the queue can have various effects on memory requirements and running time. Longer queues can load more data into memory at once, speeding up training or inference but increasing memory usage. In general, the optimal queue length will be determined by RAM capacities.

The queueing pipelines for training with and without data augmentation are depicted in Figure 2.10 and Figure 2.11, respectively. It is important to note that although the training and testing phases maintain complete separation between the unchanging training, validation, and test datasets, a random subset of each of them is selected in each training epoch and prediction run. The variability in the selected subsets is due to the inherent stochastic properties of TorchIO's queue. Nevertheless, considering the characteristics of the dataset used in this

project and the distribution of labels within it, there should not be any issues regarding model evaluation and comparison consistency in the validation and testing steps.

Non-augmented dataset

When training with the original image volume, a sole subject is available. To train, validate, and test in disjoint patch sets, the customized grid sampler presented in subsection 2.3.1 was adapted for the non-augmented training case. Within the valid patches, which remain unchanged for the same subject and patch dimensions, 70% of them were allocated for training, 20% for validation, and the remaining 10% for testing. The initial train-validation-test dataset split was conducted in a non-random manner to prevent highly overlapping patches from being assigned to different subsets. The division was performed among the valid patches rather than the initial volume to avoid potential issues arising from patch sizes not fitting appropriately in the smaller validation or test sub-volumes especially.

The queue method from TorchIO needs at least two subjects as input. Therefore, the first step was to split the XNH volume in two halves and treat them as two different subjects. At the beginning of each epoch, the training patch queue is shuffled.

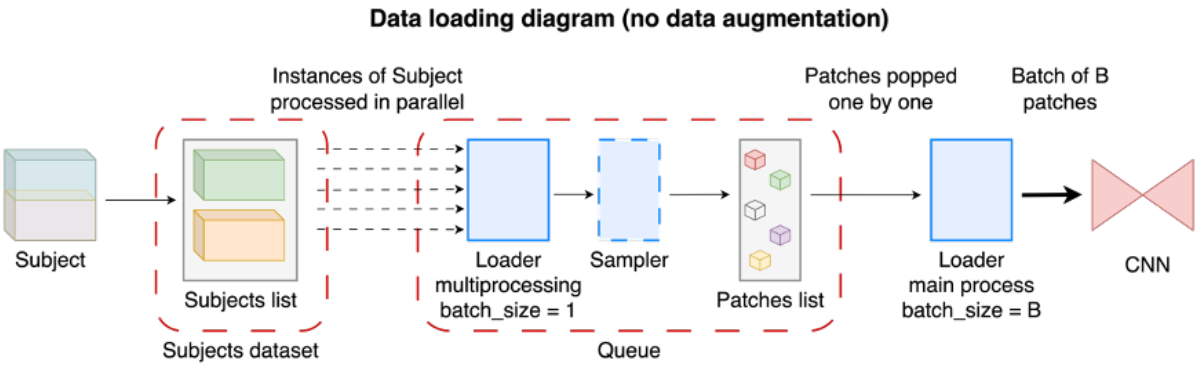


Figure 2.10: Pipeline of `torchio.Queue` for the non-augmented data loading. Figure adapted from [11].

Augmented dataset

The low image resolution and signal-to-noise ratio of the XNH volumes challenged a robust segmentation of axons with a mean diameter smaller than 2.1 µm. As a result, the estimated mean diameter and volume-weighted mean diameter were significantly larger compared to the estimates derived from diffusion MRI. Furthermore, axon trajectories exhibit greater variability in real-life scenarios beyond the 54 segmented axons analyzed in this study. Thereby, the purpose of data augmentation is to expand the dataset and assist the CNNs in recognizing and segmenting these yet unlabeled axons.

To align with TorchIO's queue pipeline, the data augmentation was performed on a per-subject basis before data loading. Recall that each subject contains the processed image, the merged segmentation mask containing labels for axonal and extra-axonal structures, and the cylindrical foreground mask. Initially, twelve copies of the subject were created, with ten copies designated for training and the remaining two for validation. The original non-transformed volume was reserved for testing, while a random transformation was applied to each of the twelve subject copies.

The selected transformations included flipping, scaling, and rotation, which were composed and each of them applied based on predefined probabilities. Furthermore, each transformation randomly selected one or more axes in which to be applied. The flip probability was set to 0.3, scaling ranged between 0.25 and 1 while maintaining isotropy, while rotation could occur at any angle. It is important to note that the transformations applied to the images and masks remained consistent within each subject. Especially, scaling and rotation are particularly useful for representing smaller axons and capture different axon trajectories.

In analogy to the non-augmented dataset case, the sampler introduced in subsection 2.3.1 was used to extract a specified number of random patches from each volume. However, unlike in the non-augmented dataset case, the valid patches did not need to be divided into train, validation, and test subsets, since the splitting was conducted prior on a subject-wise basis as previously explained. At the beginning of each epoch, the training subject list and patch queue are shuffled.

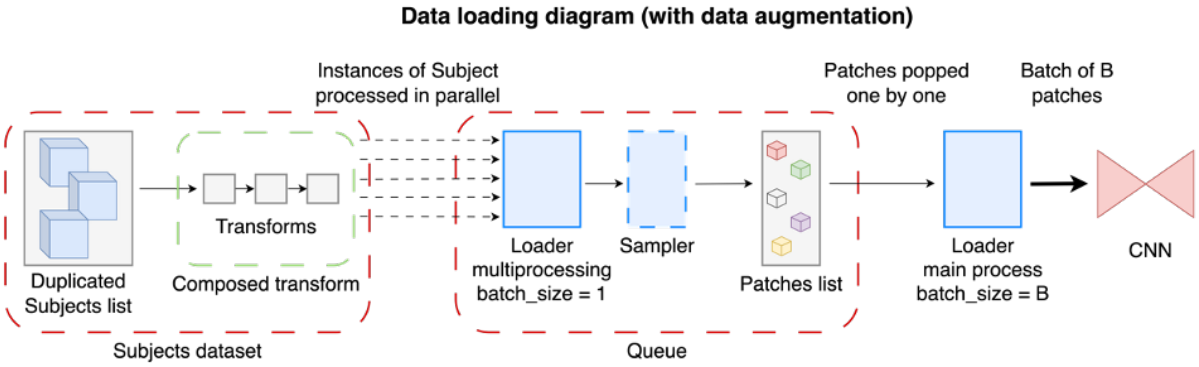


Figure 2.11: Pipeline of `torchio.Queue` for the augmented data loading. Figure adapted from [11].

2.3.3 Loss functions and performance metrics

The MONAI [4] Open Source Framework for AI Development in Medical Imaging greatly facilitated the utilization of loss functions and performance metrics for training and testing the models.

Another limiting factor in 3D medical image segmentation is class imbalance, which often leads to over-segmentation of classes with a high voxel share. This issue becomes more critical in multi-class segmentation scenarios. The merged segmentation volume exhibits clear class imbalance, with the following proportions for each labeled class across the entire volume: 0.53% for blood vessels, 1.04% for vacuoles, 3.21% for cell clusters, 1.41% for axons, and 3.01% for myelin. When considering only the labeled voxels, these proportions become 5.79% for blood vessels, 11.27% for vacuoles, 34.91% for cell clusters, 15.31% for axons, and 32.72% for myelin.

Taking class imbalance into account when calculating performance losses and metrics is crucial for adequately evaluating the model. Failure to do so may result in the model struggling to learn the features and characteristics of the underrepresented classes and can lead to a bias in the training process, where the model is more likely to predict the majority classes, resulting in poor performance in the minority classes.

During training and validation, the main focus was on the Dice loss (Equation 2.5) and the voxel-wise Cross Entropy (CE) loss (Equation 2.6) for multi-class classification, with subsequent modifications. The Dice coefficient quantifies the overlap between the prediction map and the ground truth map, yielding a value between 0 and 1. A score of 1 indicates perfect and complete overlap. The Dice calculation was smoothed by adding a small constant to both the denominator to prevent division by zero, and in the numerator to avoid zero values. The CE loss quantifies the discrepancy between the class predictions and the target classes for each voxel in the image volume, and then averages over all voxels.

The multi-class Dice loss can be defined as:

$$\text{Dice Loss} = 1 - \frac{2 \times \sum_{i=1}^{C} \text{TP}_i}{2 \times \sum_{i=1}^{C} \text{TP}_i + \sum_{i=1}^{C} \text{FP}_i + \sum_{i=1}^{C} \text{FN}_i} \tag{2.5}$$

where C is the total number of classes and for each class i: TP_i is the number of true positive predictions, FP_i is the number of false positive predictions, and FN_i is the number of false negative predictions.

The multi-class CE loss can be expressed as:

$$\text{Cross Entropy Loss} = -\frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{C} (y_{ji} \log(\hat{y}_{ji})) \tag{2.6}$$

where C is the total number of classes, N is the total number of voxels in the image and for each class i and voxel j: y_{ji} is the binary ground truth label for voxel j in class i (one-hot encoding), $\hat{y}_{ji}$ is the predicted probability of voxel j belonging to class i, and log is the natural logarithm function.

In the initial stage of the experimental phase for the current study, both losses individually, the generalized Dice loss, and a masked and combined version of each were explored. Subsequently, the focus shifted to the masked generalized Dice loss due to its outstanding performance as it described in detail in subsection 3.1.1. The Dice loss class from MONAI and the CE loss from PyTorch calculate the batch-wise average losses between the ground truth and the predicted segmentation. Both losses support multi-class segmentation. To address the problem of class imbalance, the generalized Dice loss was considered because of its class re-balancing ability and its effectiveness as a deep learning loss function for tasks involving imbalanced data.

Additionally, an approach involving masking the Dice loss and the generalized Dice loss was employed to address the issue of sparse data annotation and enable the model to identify initially unlabeled elements. By using the non-masked version of these losses, the unlabeled voxels (labeled as 0) were considered as background, which posed challenges for the models to accurately segment non-labeled axons and extra-axonal structures, due to the features learned from these voxels. Excluding the unlabeled class from the calculation, i.e., setting the weights of unlabeled voxels to zero, the model learned only from labeled voxels, resulting in improved generalization across the entire volume.

In general, MONAI provides the option to exclude the background class in almost all of its loss methods, leaving out from the loss calculation all the unlabeled voxels from both the ground truth map and the predicted mask. However, this method missing information from the wrongly unlabeled regions in the prediction map, which were labeled in the ground truth, leads to prefer the choice of a masked loss. The merged original segmentation mask was turned binary with unlabeled (0) and labeled (1) identifiers (Figure 2.12), and it was used to mask the loss taking into consideration the unlabeled masked voxels in the predicted segmentation. Dilating the mask could be beneficial in further research for detecting boundaries between elements within the mask and the unlabeled background. Various modifications to the source code from MONAI losses were carried out to meet the requirements of the study.

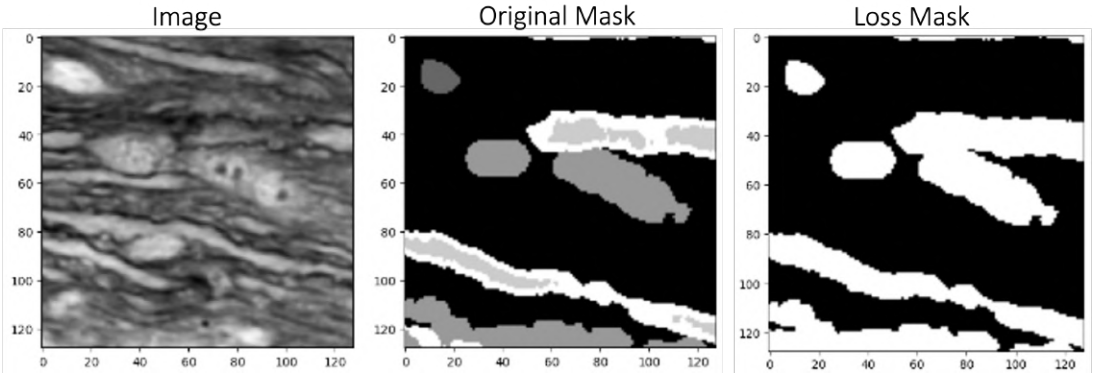


Figure 2.12: Example of the generated mask utilized for loss computation. The foreground (1) voxels are considered in the loss calculation, while the background (0) voxels are excluded.

The effects of masking the loss were tested and depicted using the F1-score (Equation 2.7), which combines precision and recall to provide a balanced measure of the model's accuracy. It ranges from 0 to 1, a higher score indicating a better performance, and provides an overall evaluation of the classification performance.

$$\text{F1-score} = (2 * \text{Precision} * \text{Recall})/(\text{Precision} + \text{Recall}) \tag{2.7}$$

with

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

where TP is the number of true positive predictions, FP is the number of false positive predictions, and FN is the number of false negative predictions. In multiclass problems, the F1-score is computed for each class individually, and then averaged to obtain an overall F1-score.

A weighted combination of the Dice loss and the CE loss could enhance segmentation results for minority classes by simultaneously maximizing the Dice similarity coefficient and optimizing the class probability of the predicted segmentation, to better align the ground truth segmentation and address class imbalances. Nonetheless, including the CE loss may not be the best option for sparsely annotated data, following the argumentation in the previous paragraphs. Note that the application of masks in the loss functions is limited to the Dice loss. Therefore, when employing a masked combination of Dice and CE loss, the mask is applied to the Dice loss exclusively, while the CE loss is computed over the whole patch. This distinction arises from time constraints within the scope of this project and the absence of a pre-implemented masked CE loss in the available MONAI resources.

Every loss takes as input the probabilities output from the network. For testing, the masked generalized Dice loss was used in order to compare the performance of different models. Nevertheless, when evaluating the model performance on entire volumes as a last step, where each voxel belongs to a different class, the Dice loss from `torchmetrics` was employed, which accepts label input.

2.3.4 Other settings

Adam Optimizer

The optimizer used during training is the Adam optimizer with a learning rate of 0.001. The suitability of this learning rate is supported by the steepness and shape of the average train loss curve depicted in Figure 2.13 and similar behavior observed in the remaining experiments that were carried out, which will be explored in detail in chapter 3. The Adam [6] optimizer computes individual adaptive learning rates for different parameters from estimates of the first and second moments of the gradients, which is beneficial for capturing fine details and complex structures within the image and handles sparse gradients commonly encountered in image segmentation tasks, ensuring stable and effective optimization. Additionally, hyper-parameters have intuitive interpretations and typically require little tuning.

Early stopping

The number of training epochs was set to one hundred. To prevent overfitting on the training set, an early stopping strategy [7] with a patience of 7 was utilized. The training process monitored the validation loss, and if the loss remained unchanged for several consecutive epochs, the training stopped. A checkpoint of the model was saved each time the validation loss decreased, ensuring that the best model was retained at the end of the training process.

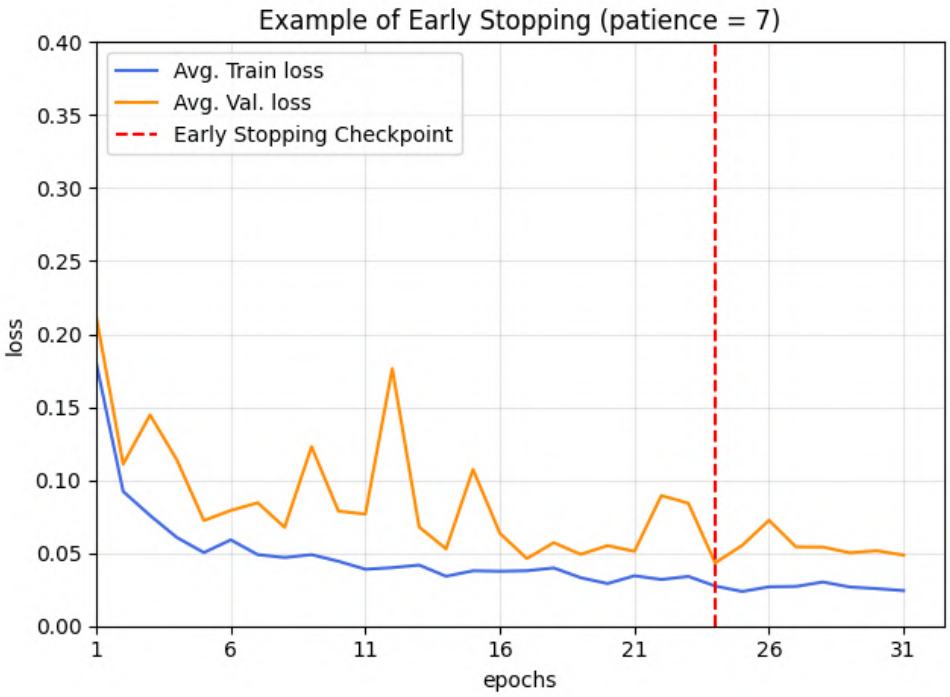


Figure 2.13: Illustration of early stopping for the 3D U-Net model presented in Figure 2.7 with patch size of $128 \times 128 \times 128$ and masked generalized Dice loss.

TorchIO's grid aggregator

The grid aggregator offered by TorchIO framework played a crucial role in facilitating dense inference. This class enabled to re-build the complete 3D image volume from patches after inference of batches extracted by a grid sampler. Moreover, the patch overlap mode can be customized to either crop, average, or weight the overlapping predictions. The overlap selected for this project was 10 voxels.

After testing the three overlapping mode options offered by TorchIO (crop, average, and 'hann'), the cropping mode was selected as the most suitable choice. The averaging method incorrectly classified many voxels as axons in the overlapping sections, resulting in a grid-like segmented area labeled as axon, as depicted in Figure 2.14a. On the other hand, when applying the 'hann' weighted method, the final predicted volume exhibited excessive noise, as illustrated in Figure 2.14b. However, the cropping mode effectively handled the overlapping sections as observed in Figure 2.14c, providing satisfactory results overall.

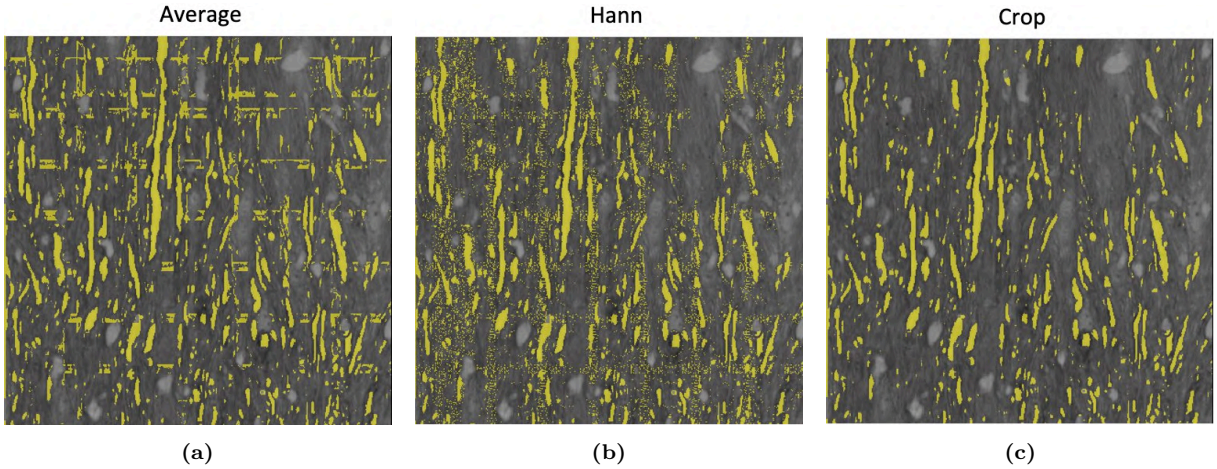


Figure 2.14: Performance of overlap modes in grid aggregator. (a) Average mode: predictions in overlapping areas are averaged with equal weights. (b) Hann mode: predictions in overlapping areas are weighted using a Hann window function. (c) Crop mode: overlapping predictions are cropped. The predictions were generated using the 3D U-Net model described in Figure 2.7 with a patch size of $64 \times 64 \times 64$ voxels, data augmentation, and masked generalized Dice loss.

2.4 Experimental Setup

For developing this project the LSF 10 Cluster from the High Performance Computing (HPC) services provided by DTU Computing Center (DCC) were used. Initially, CPU was employed. However, to enhance processing speed, the utilization of GPUs became necessary. Specifically, jobs were submitted to the LSF10-setup, allocating resources from the `gpua100` and `gpuv100` queues, utilizing the NVIDIA A100 and V100 GPUs, respectively. The code was compiled with CUDA 11.7 for nodes with GPU.

In this project, `neptune.ai` [8] platform was employed for logging and visualizing metrics and parameters, and storing outputs and models generated during the experiment runs. The use of `neptune.ai` facilitated experiment tracking and management, enabling effective organization

and analysis of the results. The integration of `neptune.ai` in the code streamlined the logging process and enhanced the arrangement and reproducibility of the project.

Python implementation and the complete source code developed for the project can be accessed on GitHub: https://github.com/MirenLurBarquin/msc_thesis.

Chapter 3

Results

This chapter explores the different models described in chapter 2, highlighting the parameters and features that enhance performance and boost the segmentation of different elements in the image, with a special focus on axons. Throughout the chapter, results obtained during training and testing will be presented, including relevant metrics or performance measures. Images, learning curves, F1 scores, loss trends, and other evaluation charts will be utilized to illustrate the later decision-making process.

Initially a baseline model will be studied, followed by the incorporation of data augmentation. Additionally, the concept of 'cross-hair' filter will be explored. Lastly, a comprehensive analysis of the diameter and length of the predicted axonal structures will be presented.

3.1 Non-augmented baseline U-Net model

The initial model to be studied is the U-Net CNN described in subsection 2.2.1, which served as the baseline model in the former sections of this dissertation. The dataset for this case study consists solely of the original XNH image volume and ground truth segmentation, without any transformation applied.

3.1.1 Patch size and train loss

The first experiment is to explore various patch sizes and examine the different losses introduced in subsection 2.3.3. The explored patch sizes are $64 \times 64 \times 64$ and $128 \times 128 \times 128$ voxels, following the reasoning described in subsection 2.3.1. The training losses tested can be found in the legend of Figure 3.1, which comprise a combination of masked and non-masked Dice and CE losses.

For testing purposes, the masked generalized Dice loss was selected to evaluate the overlap performance on the labeled voxels. This selection was made to avoid penalizing newly labeled voxels that were not initially labeled, as previously mentioned in chapter 2. Additionally, every test in this section was carried out for a batch size of 6 patches and a dataset composed of 560 patches for training, 160 for validation, and 80 for testing, following a common 70-20-10 percent train-validation-test data split.

The results of the model evaluation are presented in Figure 3.1 and Table 3.1. The plot displays the losses for patch sizes of $64 \times 64 \times 64$ and $128 \times 128 \times 128$ voxels represented by dotted and continuous lines, respectively. It is important to note that comparing the Dice loss of models trained on different patch sizes may not provide a reliable measure of their performance due to the difference in spatial resolution and captured contextual information in each case. Nonetheless, both the models trained and evaluated in the smaller patches and the ones using a larger patch size suggest a better performance with the masked generalized Dice loss. With this loss, the model trained and evaluated on $128 \times 128 \times 128$ voxel patches maintained an average test loss per batch below 0.14, reaching values below 0.06. Similarly, for the model trained and evaluated on $64 \times 64 \times 64$ voxel patches, the average test loss per batch remained below 0.25, with values below 0.12.

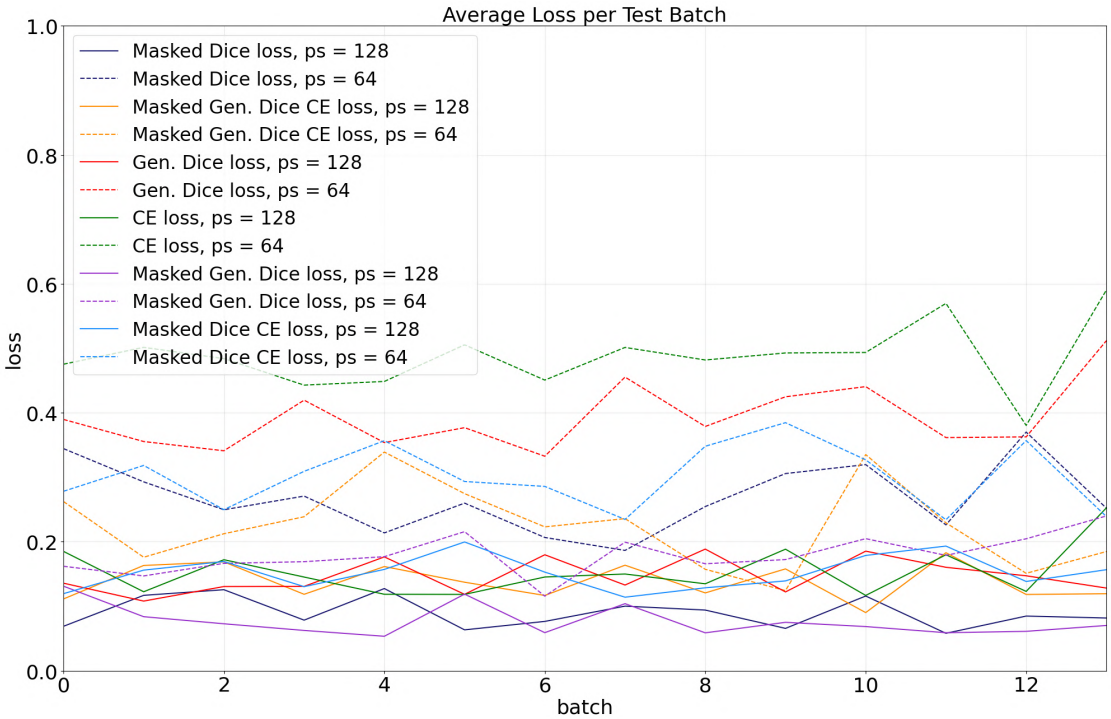


Figure 3.1: Average masked Dice loss per batch in the test set comprising patches of dimensions $128 \times 128 \times 128$ and $64 \times 64 \times 64$ for each corresponding model. 'Gen.' stands for Generalized and 'ps' for patch size. The losses and patch sizes were evaluated for the 3D U-Net presented in subsection 2.2.1 without data augmentation.

Loss	Average Test Loss	
	Patch Size	
	128	64
Masked Dice	0.0898	0.2681
Masked Gen. Dice CE	0.1379	0.2246
Gen Dice	0.1461	0.3932
CE	0.1538	0.4870
Masked Gen. Dice	0.0770	0.1799
Masked Dice CE	0.1525	0.3012

Table 3.1: Average masked Dice loss in the test set. Same model configurations as the ones used for Figure 3.1.

To provide a better visualization of the model's performance with each train loss function, Figure 3.2 presents various examples showcasing different test patch predictions. Additional test prediction examples and larger figures can be found in Appendix A. Based on the predicted image outputs, several observations can be made. For a better visualization of the results, a performance colored mask was generated. This mask assigns colors to voxels as follows: green for correctly classified voxels, blue for the voxels labeled in the ground truth but not in the prediction, red for mislabeled voxels, and yellow for the newly labeled voxels. Note that the voxels colored in yellow do not necessarily indicate a correct classification.

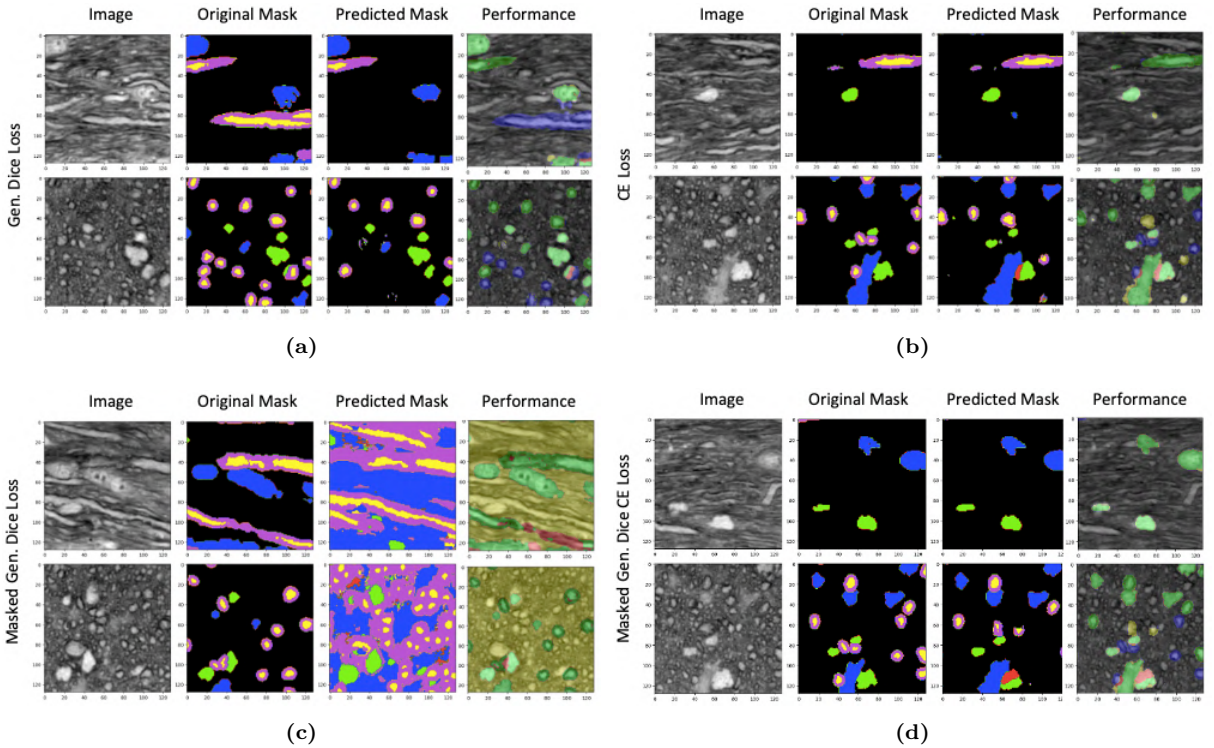


Figure 3.2: Examples of predictions from the test set using different loss functions. Models trained in patches of dimensions $128 \times 128 \times 128$ voxels. (a) Generalized Dice Loss, (b) Cross Entropy Loss, (c) Masked Generalized Dice Loss, and (d) Masked Generalized Dice Cross Entropy Loss.

Both the Generalized Dice loss and the CE loss demonstrate effective prediction capabilities for elements belonging to different classes. However, they exhibit limitations in accurately capturing certain axonal and extra-axonal elements present in the unlabeled background. Moreover, the presence of unlabeled voxels persists, as the network predominantly learns to classify these voxels as background, rather than accounting for the sparsely annotated data. By using a masked loss for training, such as the masked generalized Dice loss in Figure 3.2c, new labels are assigned to the initially unlabeled background voxels, allowing the detection of additional elements in the sparsely annotated image and providing a more dense segmentation. Consequently, it enables the segmentation of a larger number of axons compared to the 54 present in the ground truth map. The test loss is still measured considering solely voxels labeled in the original segmentation mask. Figure 3.2d illustrates that combining the masked generalized Dice loss with the unmasked CE loss, already results in the omission of elements in the unlabeled background. Thus, for this study, the use of masked losses is essential in order to achieve a dense segmentation of the sample.

In Figure 3.3, the non-masked loss exhibits a good balance between precision and recall in the classification model, while the masked case shows the opposite result. It is attributed to the fact that with the masked loss the ground truth's unlabeled background voxels rarely remain unlabeled in the predicted output. This plot supports the observations made in Figure 3.2.

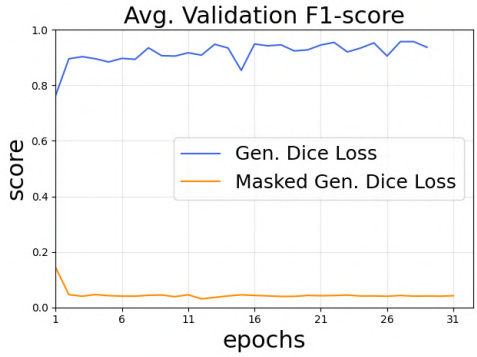


Figure 3.3: Validation phase results: average F1-score per epoch for the U-Net model presented in subsection 2.2.1 without data augmentation and employing masked and non-masked generalized Dice losses.

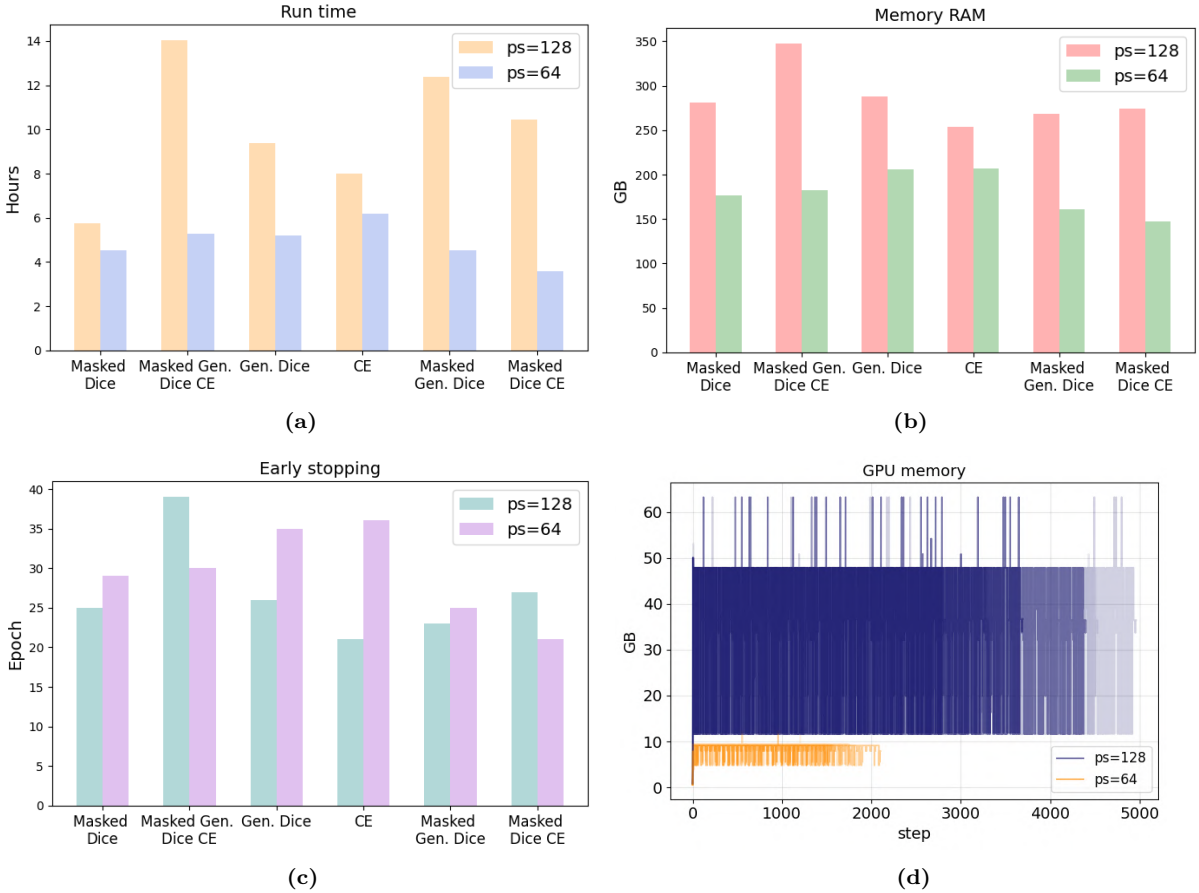


Figure 3.4: Comparative analysis of patch sizes 128 and 64. Comparison of the key monitoring metrics of the baseline 3D U-Net network for each loss. (a) Total run time. (b) Total RAM used. (c) Early stopping epoch. (d) GPU memory requirements; different blue hues represent patch size 128 models with different training losses while varying orange hues correspond to patch size 64 models.

Additional insights on runtime, early stopping epoch, and memory usage can be found in Figure 3.4. When employing a $64 \times 64 \times 64$ voxel patch size, as shown in Figure 3.4a, a substantial decrease in the total train and evaluation time is evident compared to using $128 \times 128 \times 128$ dimensions. Similarly, the visualization in Figure 3.4b demonstrates a significant reduction in RAM requirements for the smaller patch case. The convergence of the models does not appear to rely directly on the patch size, as there is no notable difference in the early stopping epoch between the two cases (see Figure 3.4c). The GPU memory requirements are considerably higher for the larger patch size as depicted in Figure 3.4d.

The increased memory and computation time requirements are a direct consequence of the fact that the 3D U-Net model being used in this initial approach involves 2154 MB forward/backward passes in the $64 \times 64 \times 64$ voxel patch size case, while in the case of a $128 \times 128 \times 128$ voxel patch size, they require 17232 MB. Smaller patch sizes result in fewer elements to store, thereby reducing memory requirements. Moreover, as the patch size increases, the amount of memory needed to store intermediate results, gradients, and activation maps increases, requiring higher GPU resources. Models with smaller dimension inputs tend to exhibit improved computational efficiency at processing and updating weights.

To conclude this section, the loss function for training the subsequent experiments in this thesis is the masked generalized Dice loss, which has shown superior performance. This loss function effectively handles class imbalance and considers only voxels labeled in the ground-truth segmentation, enabling to detect unlabeled axonal and extra-axonal elements in the image. Note that this is the same loss used for testing the model performance.

The patch size will be further investigated in the upcoming subsections on an ongoing basis. This analysis will be based on testing the performance of each class in entire volume predictions rather than patch-wise assessments.

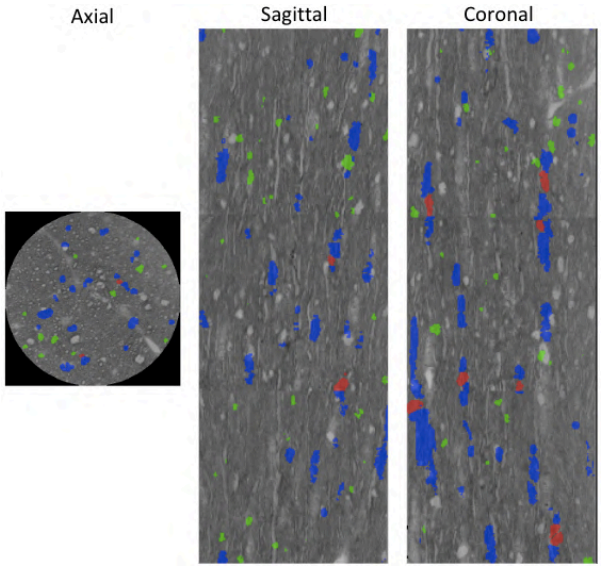
3.1.2 Combined volume segmentation

In the non-augmented case presented in this section, the fourth segment of the combined volume from Figure 2.1 was used for training and testing. Now, the performance of the U-Net model will be visualized in the combined volume containing the remaining three sub-volumes. This combined volume has dimensions of $410 \times 410 \times 1158$ voxels, which corresponds to 153.6 µm diameter and 443.5 µm height. No ground-truth segmentation is available for the axons in this prediction.

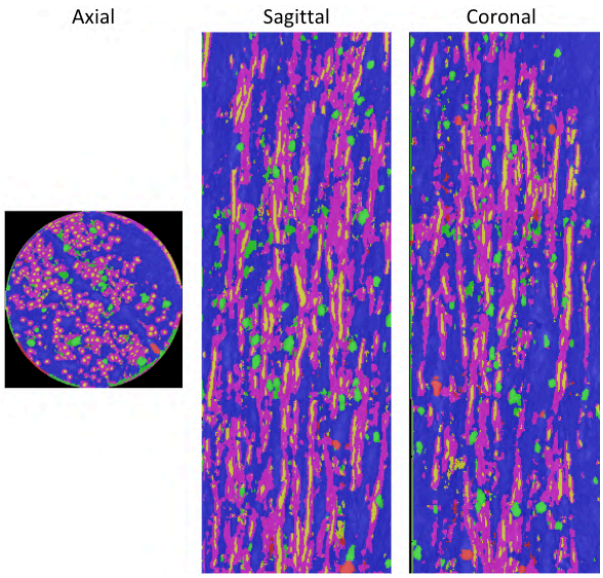
To achieve a dense segmentation of entire volumes in this and the subsequent sections, the aggregator function from TorchIO will be utilized, specifically employing the crop overlap mode, as introduced in subsection 2.3.4. Once the predicted volume is obtained, the first step involves unlabeling every background voxel, which refers to voxels not belonging to the cylindrical shape of the original sample.

This subsection analyzes and compares the predicted combined volumes generated by the U-Net CNN model with masked generalized Dice loss and no data augmentation, for the two different patch sizes being studied. For an effective result comparison, a pairwise examination of the different available output views will be carried out along various figures.

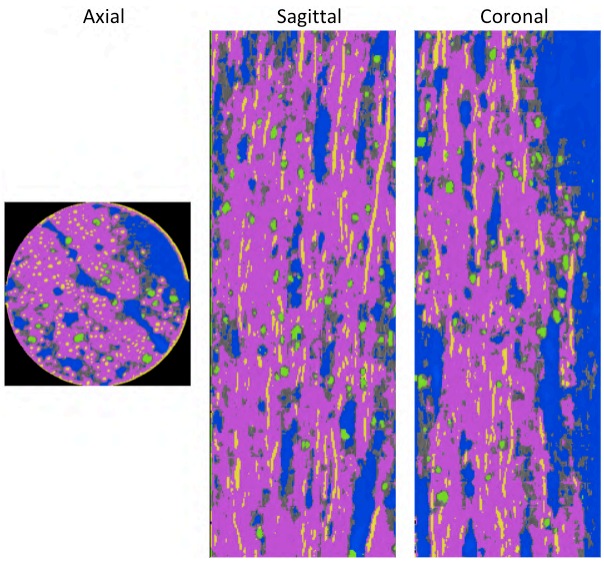
In Figure 3.5, it can be observed how for a larger patch size the segmentation of the myelin is refined and the cell cluster label dominates the background. Conversely, the smaller patch size leads to a greater presence of myelin. Moreover, for patch size of dimensions $64 \times 64 \times 64$ voxels, a portion of the voxels remains unlabeled (13.88%), whereas this is not the case for a patch size $128 \times 128 \times 128$ voxels (0.00% of unlabeled voxels). Confusion matrices in Figure 3.6 confirm these findings, demonstrating that larger patch sizes successfully label previously unlabeled voxels mainly as cell clusters, while smaller patch sizes leave some voxels unlabeled. Importantly, recall that there is no available ground truth segmentation of axons and myelin for this combined volume. Therefore, these classes have not been included in the confusion matrices. Nevertheless, they have been accounted for their normalization to maintain a consistent evaluation of the segmentation.



(a) Ground truth

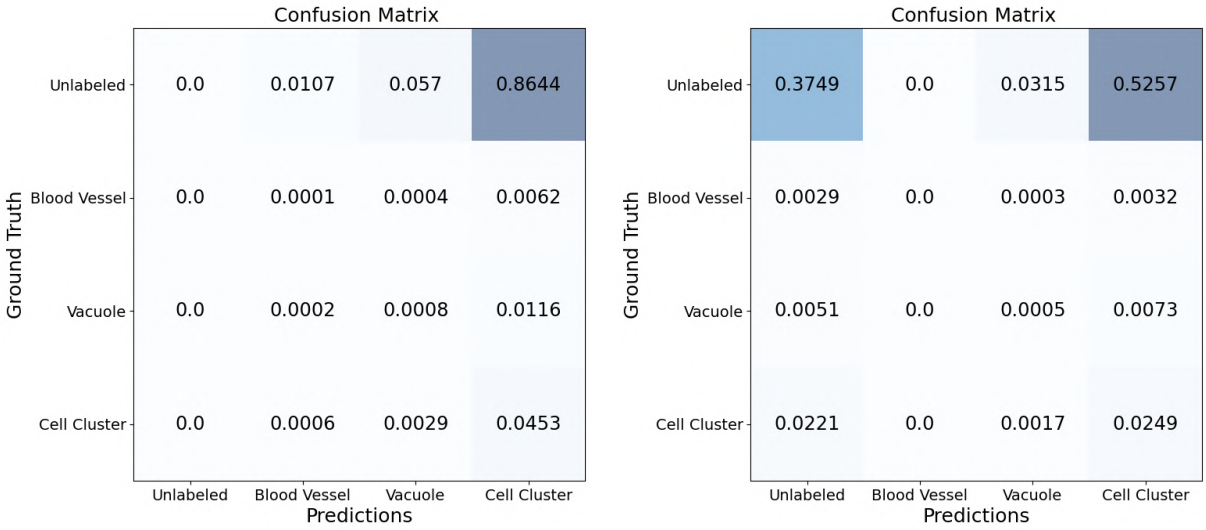


(b) Patch size = $128 \times 128 \times 128$



(c) Patch size = $64 \times 64 \times 64$

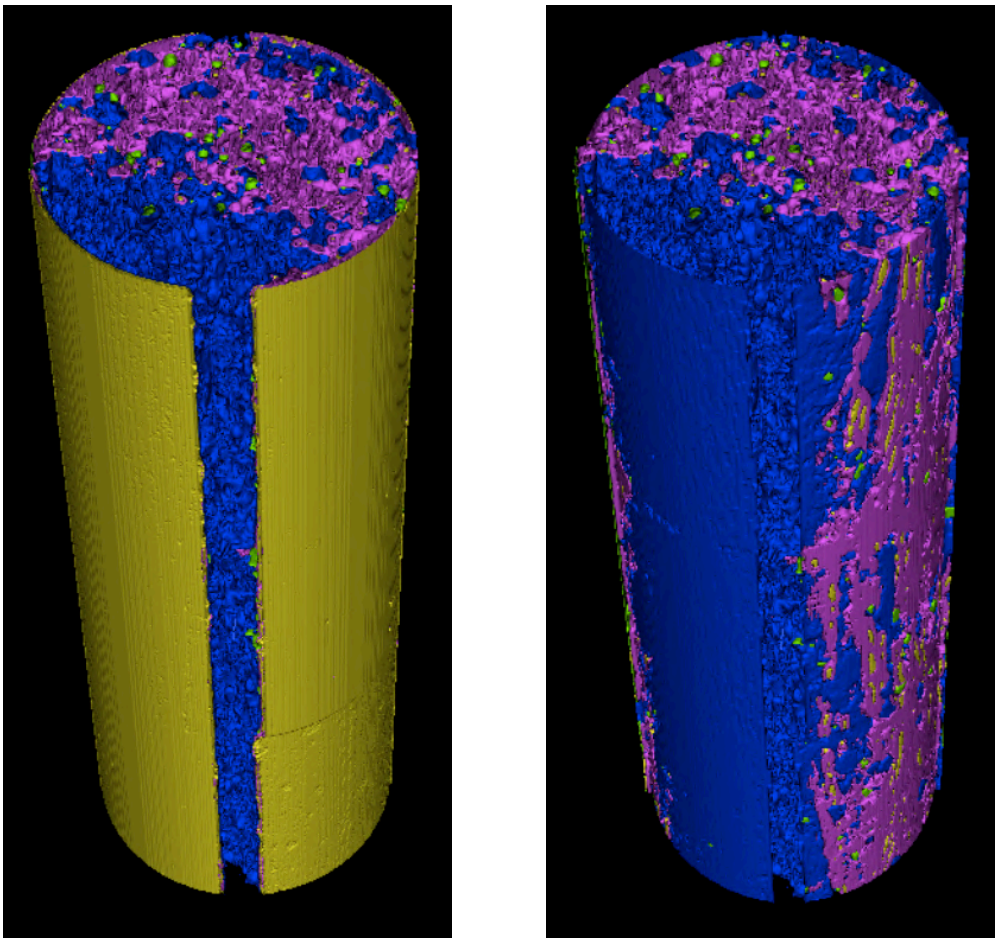
Figure 3.5: Segmentation of, blood vessels (red), vacuoles (green), cell clusters (blue), axons (yellow), and myelin (magenta) visualized in ITK-Snap.



(a) Patch size = $128 \times 128 \times 128$

(b) Patch size = $64 \times 64 \times 64$

Figure 3.6: Confusion matrices for the whole predicted volume, normalized over the entire combined volume.



(a)

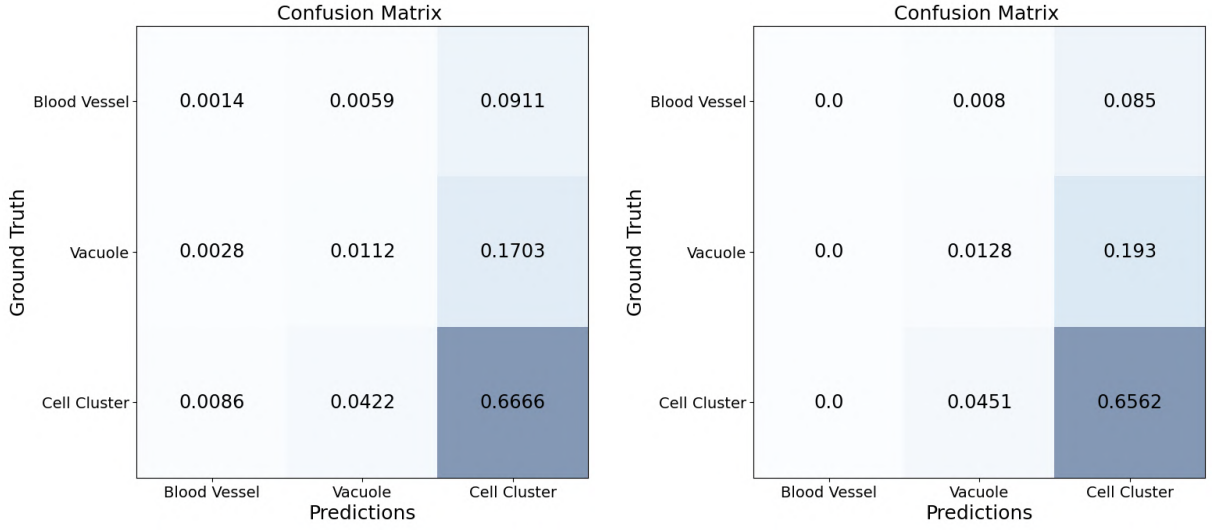
(b)

Figure 3.7: (a) Predicted volume and (b) refined predicted volume. The segmentation corresponds to model training and evaluation conducted with a patch size of $64 \times 64 \times 64$.

In both cases, the axial view reveals a consistently labeled ring along the cylindrical boundary, predominantly corresponding to axons and vacuoles. This feature is even more pronounced in the 3D view shown in Figure 3.7a. To facilitate the analysis and visualization of results, the cylindrical foreground mask was eroded to generate a reduced version of it, allowing to refine the segmentation map by excluding voxels within the cylindrical boundary, as depicted in Figure 3.7b. This image post-processing step will be performed for every predicted volume henceforth in this dissertation, without explicitly mentioning it.

To enhance visualization, Figure 3.9 provides a 3D depiction of each label individually. When comparing the segmentation results between the models trained and evaluated with patch sizes of $64 \times 64 \times 64$ and $128 \times 128 \times 128$ voxels, a noticeable difference is observed in their ability to capture blood vessels. This disparity is expected due to the dimensions of the blood vessels, which range from 4 to 10 µm in diameter [2] and exhibit significantly larger volumes compared to other structures. Consequently, smaller patches such as $64 \times 64 \times 64$ may not be able to capture enough intensity and shape information to distinguish these structures from the background or other elements. Furthermore, during the analysis of vacuoles and cell clusters, it was observed that the smaller patch size captures a lower number of elements in comparison to the larger patch size. As a result, the output of the smaller patch sizes exhibits a denser myelin segmentation, whereas the larger patch size does so for the blood vessel, vacuole, and cell cluster classes. In any case, post-processing steps based on the microstructural characteristics of each element class are necessary to eliminate prediction noise.

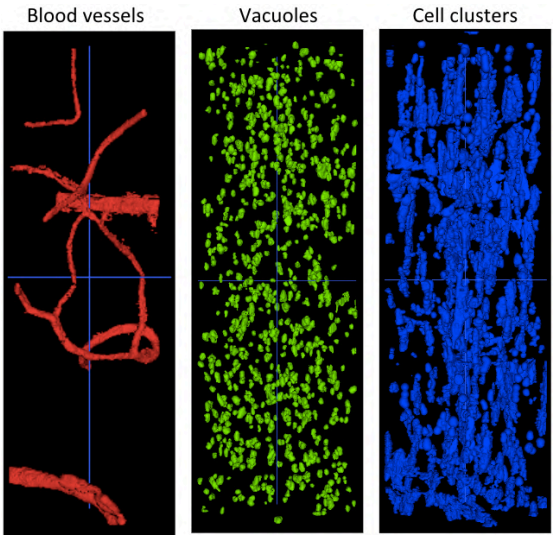
Nevertheless, the confusion matrices in Figure 3.8 reveal comparable classification performance between the two cases for voxels which were originally labeled in the ground truth segmentation map. The myelin segmentation will remain out of focus since it was generated as a helper class for axon segmentation, and lacks professional manual annotation for comparison. Overall, no significant performance differences were observed in the visual inspection of axons as a whole class. Consequently, further analysis using blob analysis and post-processing methods will be conducted specifically for the axonal structures.



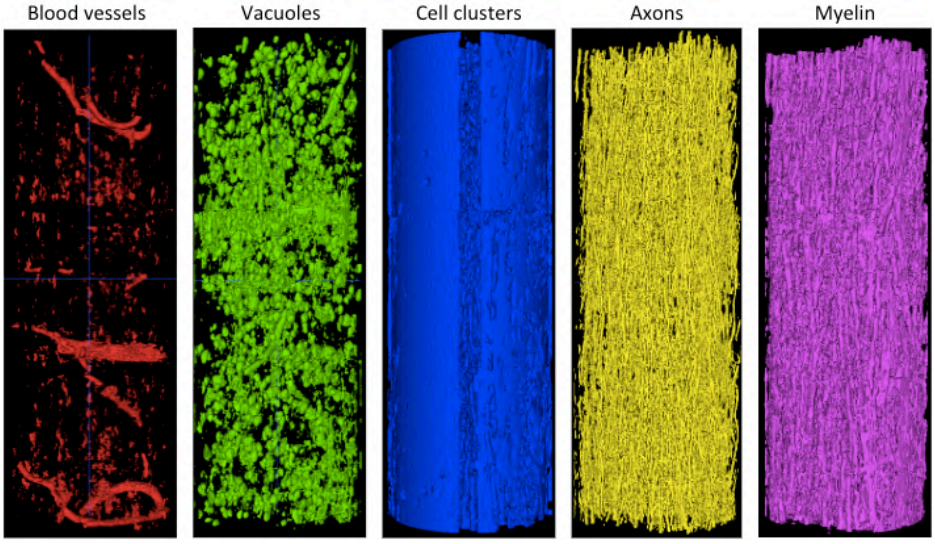
(a) Patch size = $128 \times 128 \times 128$

(b) Patch size = $64 \times 64 \times 64$

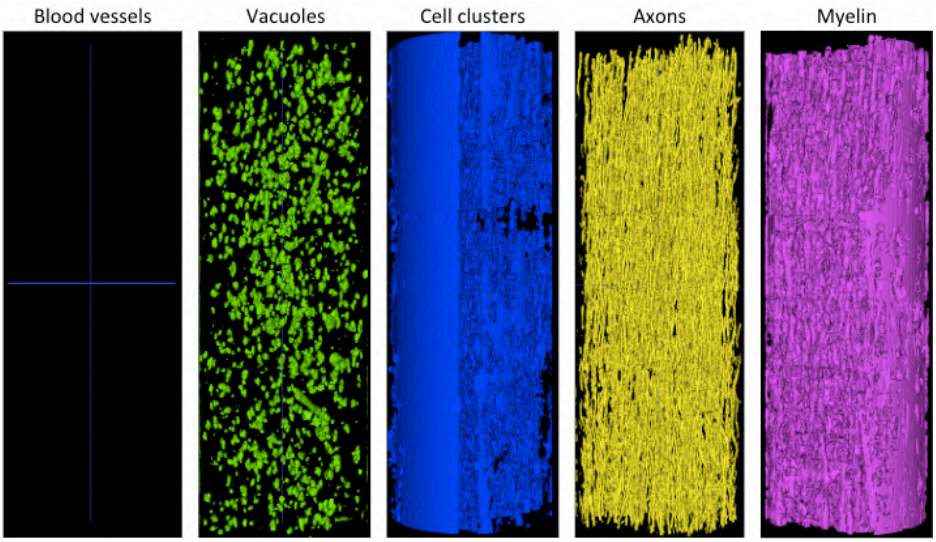
Figure 3.8: Confusion matrices for foreground voxels in the ground truth segmentation, normalized over the entire combined volume.



(a) Ground truth



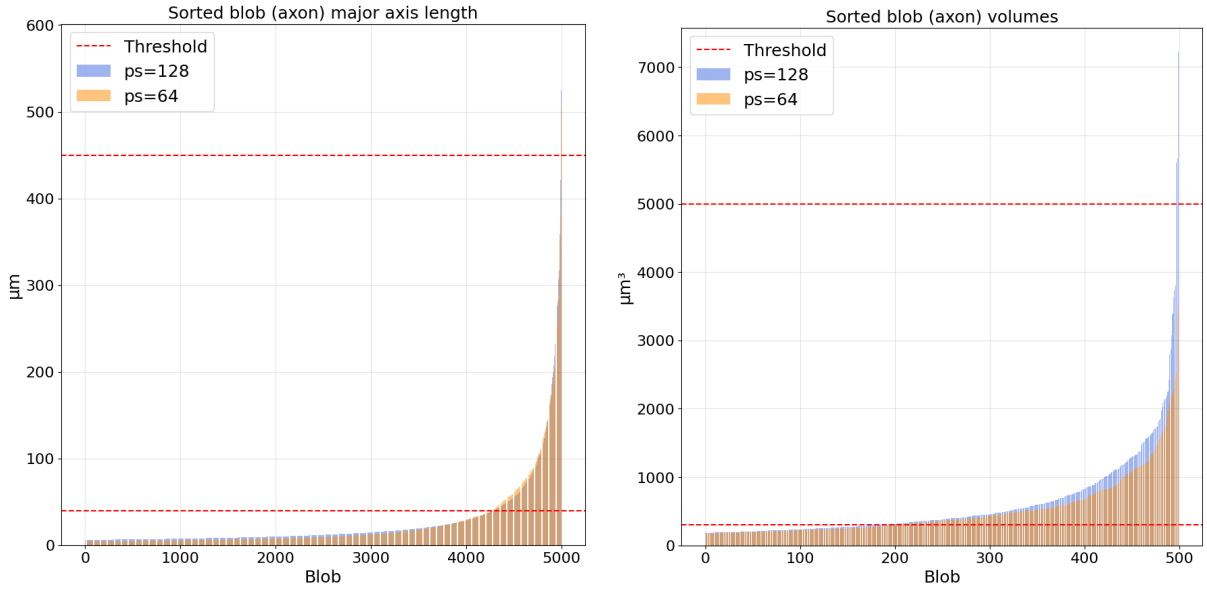
(b) Patch size = $128 \times 128 \times 128$



(c) Patch size = $64 \times 64 \times 64$

Figure 3.9: Individual 3D visualization of each segmented class.

Focusing on the axons, individual blobs within the axonal class were labeled, and their properties were measured using Python's built-in functions from the `scikit-image` image processing library. Initially, the model trained on $128 \times 128 \times 128$ voxel patches identified 46622 blobs, while the model trained on $64 \times 64 \times 64$ voxel patches identified 14536 individual structures. Nevertheless, when considering only the 5000 and 500 elements with the largest major axis length and volume, respectively, the structures present similar characteristics for both cases, with a slightly larger estimation of volumes for the model trained in the larger patch size. This comparison is depicted in the bar plots in Figure 3.10. It is important to note that the major axis length property represents the length of the major axis of an ellipse fitted to the region, i.e., it provides an estimate of the longest segment that can be drawn within the region. Thus, while this property can provide information about the elongation of a region, it may not directly correspond to the exact estimate of the axon length.



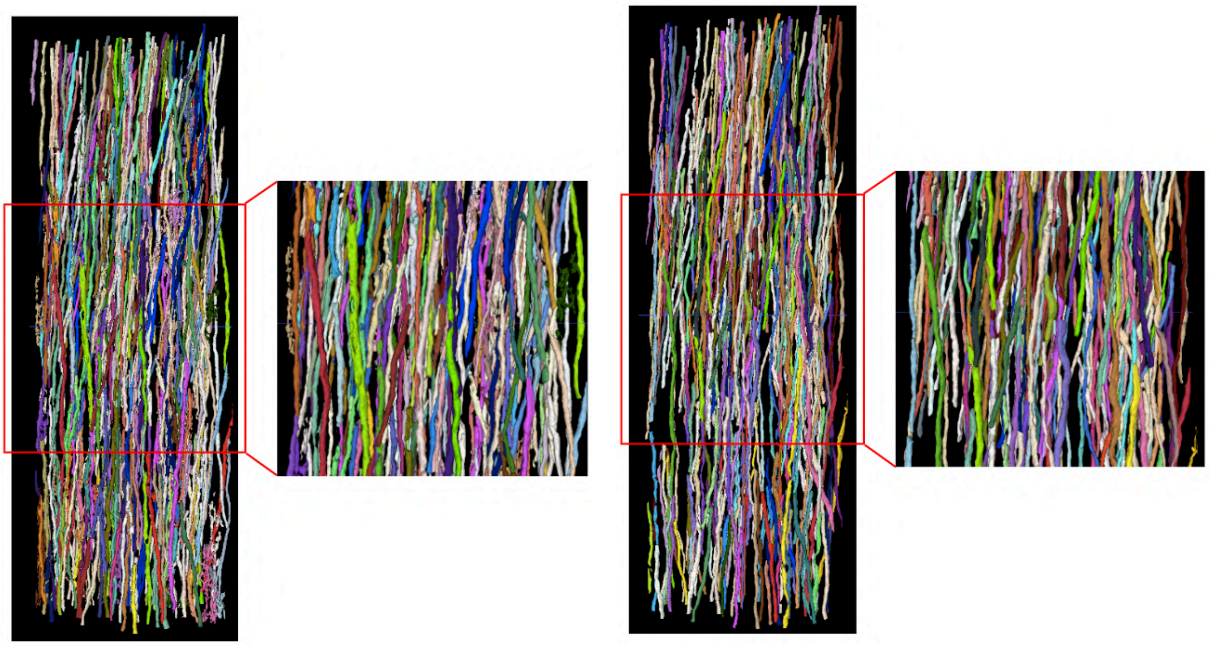
(a) **(b)**

Figure 3.10: Sorted major axis length and volumes of the greatest 5000 and 500 predicted axon segments, respectively.

To remove unwanted axonal artifacts from the predicted volume image, various thresholds were applied. Through visual inspection, lower bounds were determined for both the major axis length and volume. A lower threshold of 40 µm was set for the major axis length, while a threshold of 300 μm^3 was set for the volume. Considering that axons are thin and long tubular structures, it is reasonable to assume that the axis major length of a region can not exceed the diagonal length of the cylindrical structure within the available sample. Therefore, the upper bound for this feature was set to 450 µm. The maximum volume was thresholded to 5000 μm^3 by visual analysis of the bar plot in Figure 3.10b.

The processed predictions of segmented axons are shown in Figure 3.11. After processing, it is evident that the larger patch size produces a slightly denser axon segmentation compared to the model trained with a larger patch size. Nevertheless, the exact count on the segmented axons for each case can not be determined yet due the possibility of different shorter segments belonging to the same axon. Furthermore, it is important to consider that each axon must cross the volume longitudinally from one side to the other in out specific sample, and in any direction in volumes with crossing fibres.

Finally, the model trained for a patch size of $128 \times 128 \times 128$ voxels occasionally segments non-tube-like structures as axons, a behavior less frequently observed in models trained with smaller patch sizes.



(a) Patch size = $128 \times 128 \times 128$ **(b)** Patch size = $64 \times 64 \times 64$

Figure 3.11: Axon segmentation in the combined volume. For more detailed views of the figures, larger images can be found in figures B.1 to B.4 in Appendix B.

The upcoming section will examine the impact of data augmentation, as well as explore the training sample size and batch size.

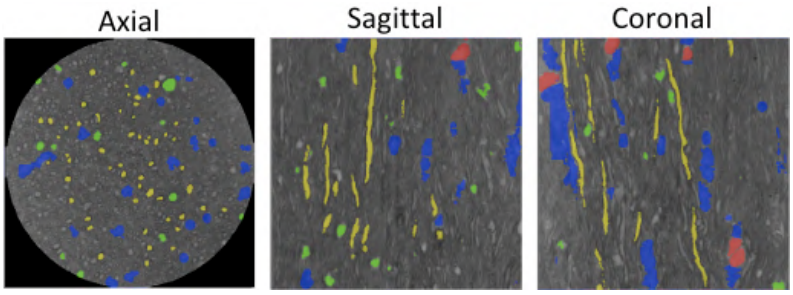
3.2 Augmented baseline U-Net model

This section focuses on evaluating the performance of the baseline U-Net model presented in subsection 2.2.1 and used in section 3.1, with the inclusion of data-augmentation. Furthermore, different batch sizes and dataset sizes will be explored. Building upon the findings from section 3.1, the masked generalized Dice loss will be employed as the training and testing loss metric.

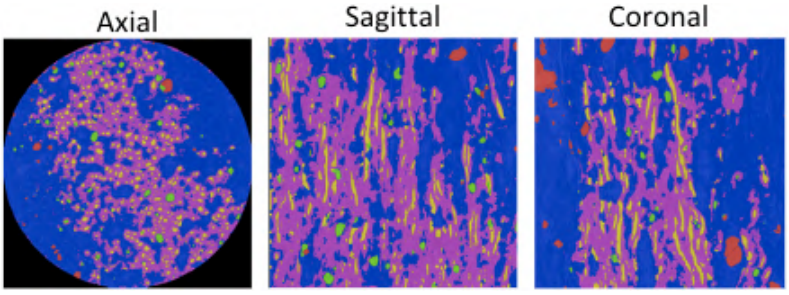
In this experiment, the model is exclusively trained on augmented data, while the original volume is reserved for testing. Ideally, non-transformed volumes would be included for training if additional sample volumes and ground truth segmentations were available. The data augmentation transformations described in subsubsection 2.3.2 were applied to augment the training dataset. The baseline model configuration for this experiment involves a dataset comprising 400 training patches, 80 validation patches, and 40 testing patches, with a batch size of 6 patches.

To retrieve the entire predicted volumes, the three available overlap modes (average, crop and 'hann') were tested in the segmentation of the original volume with patch size of $64 \times 64 \times 64$ voxels and overlap of 10 voxels. For the average mode, the masked weighted Dice coefficient yielded a value of 0.9832, for the crop mode a value of 0.9835, and 0.9779 for the 'hann' mode. These results validate the crop mode selection made in the methods chapter and previously used in subsection 3.1.2. Similarly, when predicting the model trained and evaluated with patches of dimensions $128 \times 128 \times 128$ a consistent behaviour was observed; the masked weighted Dice coefficients for the average, crop, and 'hann' modes were 0.9788, 0.9797, and 0.9733, respectively.

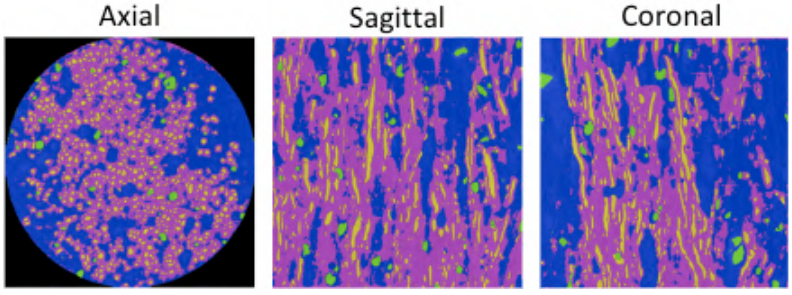
In this case, the individual sparsely segmented volume including ground truth labels for blood vessels, vacuoles, cell clusters, and axons was predicted. As in previous stages of the study, the additional myelin label was generated in an attempt to help axon class segmentation. The axial, sagittal and coronal views in Figure 3.12 reveal a denser axon segmentation for the model trained in a patch size of $64 \times 64 \times 64$ voxels. However, in the coronal view, no blood vessel (red) labels are visible in this particular case. Nonetheless, compared to the previous section's case without augmentation, the model trained with the smaller patch size produces significantly denser predictions.



(a) Ground truth



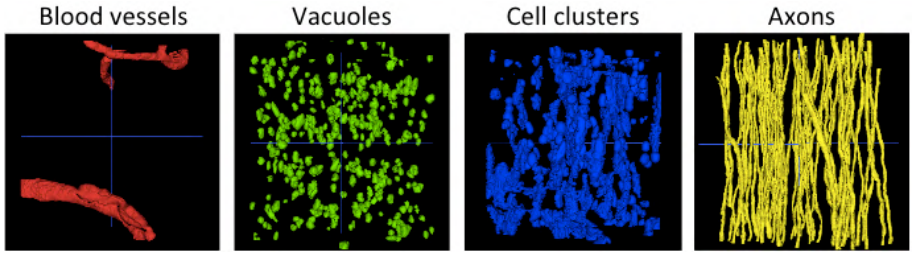
(b) Patch size = $128 \times 128 \times 128$



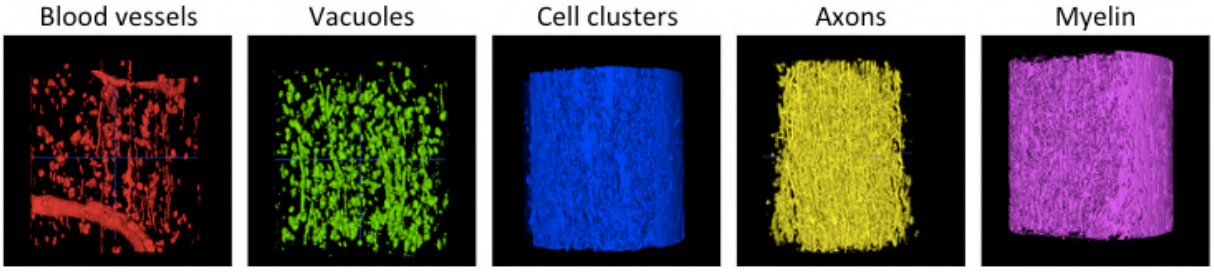
(c) Patch size = $64 \times 64 \times 64$

Figure 3.12: Prediction outputs of the augmented baseline U-Net model.

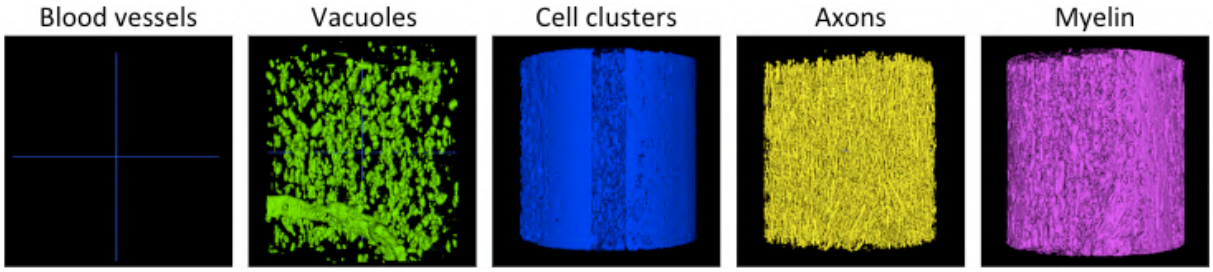
Taking a closer look at the individual segmented classes, it becomes evident in Figure 3.13b that, in the case of larger patch size predictions, certain axon segments were mistakenly identified as blood vessels or vacuoles, and some vacuoles were incorrectly classified as blood vessels. In contrast, in the case of a smaller patch size, shown in Figure 3.13c, only a minimal number of voxels (specifically, 10) were classified as blood vessels, similar to the non-augmented scenario, whereas ground-truth blood vessels were segmented as vacuoles and cell clusters in the prediction. The unlabeled voxels were predominantly segmented as cell clusters, axons, and myelin. Refer to the confusion matrix in Figure B.5 for a detailed analysis of the dense segmentation. Although the model trained with a patch size of $64 \times 64 \times 64$ performs poorly in capturing blood vessels and produces a rougher segmentation of vacuoles, it significantly improves the segmentation of the axon class by achieving a denser annotation. Furthermore, it effectively captures finer axons, which constitutes a key objective of this project and justifies the inclusion of data scaling.



(a) Ground truth



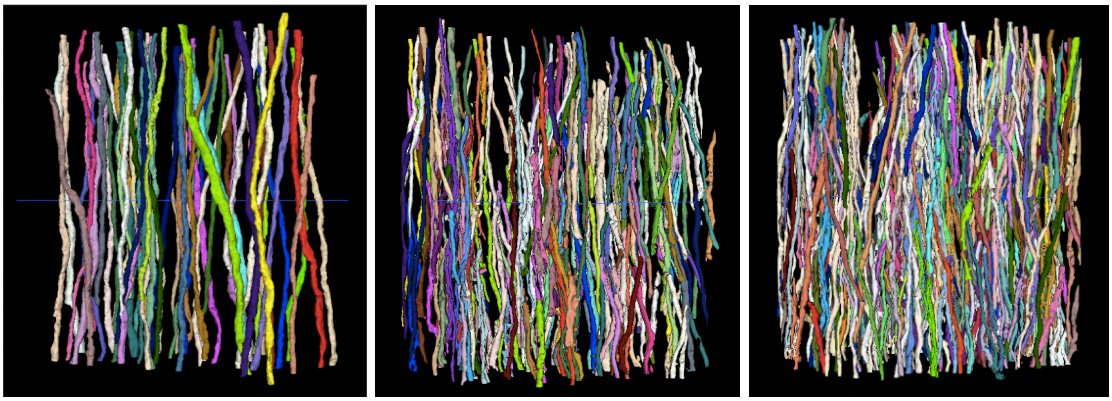
(b) Patch size = $128 \times 128 \times 128$



(c) Patch size = $64 \times 64 \times 64$

Figure 3.13: Prediction outputs of the augmented baseline U-Net model.

After labeling each axon segment individually and eliminating outlier blobs, a denser axon segmentation was confirmed during post-processing for the smaller patch size scenario. The analysis of the model trained with patch size 128 resulted in the identification of 191 axon segments, while the model trained with patch size 64 detected 305 segments after removing artifacts. Figure 3.14 illustrates the final axon segmentation results for models trained in different patch sizes. Additionally, the Dice score between the labeled ground truth voxels and the predictions was 0.98 for the model trained with the larger patch size and 0.9716 for the model trained with the smaller patch size.



(a) Ground truth **(b)** Patch size = $128 \times 128 \times 128$ **(c)** Patch size = $64 \times 64 \times 64$

Figure 3.14: Predicted axons with the augmented baseline U-Net model.

In the upcoming subsection, the analysis will focus on varying dataset and batch sizes to optimize prediction loss, as well as runtime and usage of computational resources.

3.2.1 Dataset size and batch size

Two experiments were carried out in parallel to determine the optimal dataset size and batch size, considering accuracy of the output as well as efficient utilization of time and memory resources. The baseline model used in these experiments was, as in previous sections, the U-Net model described in the methods chapter with masked generalized Dice loss, batch size of 6 patches and a dataset composed of 400 patches for training, 80 for validation, and 40 for testing the augmented scenario.

Dataset size

In the baseline model, the dataset size was established by randomly selecting 40 patches from each subject in the train, validation, and test sets. Building upon this approach, subsequent models were trained using varying numbers of patches per volume, specifically 20, 40, and 80 patches. For each case, the tests were conducted both for patches of size $64 \times 64 \times 64$ and $128 \times 128 \times 128$ voxels. Additionally, an experiment was conducted with 160 patches per volume for the smaller patch size case, aiming to capture equivalent amount of total spatial information as the larger patch size.

Samples per Volume	Average Test Loss		Runtime		RAM (GB)	
	Patch Size		Patch Size		Patch Size	
	128	64	128	64	128	64
20	0.2285	0.1432	5h 18m	2h 19m	213.85	68.89
40	0.3188	0.2427	9h 11m	3h 44m	373.20	114.48
80	0.2048	0.1449	15h 45m	6h 32m	876.81	209.76
160	-	0.2320	-	14h 58m	-	408.67

Table 3.2: Average test loss, runtime, and maximum RAM utilized in each experiment for patch dimensions of $128 \times 128 \times 128$ and $64 \times 64 \times 64$ voxels.

The results gathered in Table 3.2 show that a smaller dataset achieves lower loss compared to the baseline model, and yields similar results to a much larger dataset. Additionally, there are noticeable differences in terms of runtime and memory requirements. Furthermore, the learning curves depicted in Figure 3.15 and Figure 3.16 confirm effective training and validation procedures, which remain consistent across different dataset sizes. Despite requiring larger runtime, in general, the models trained on larger datasets converge faster and stop the training at an earlier stage than the models trained on smaller dataset sizes. This results from their improved ability to learn the underlying patterns in the data, due to the incremented data available to learn from. However, smaller dataset sizes achieve similar outputs and require less runtime.

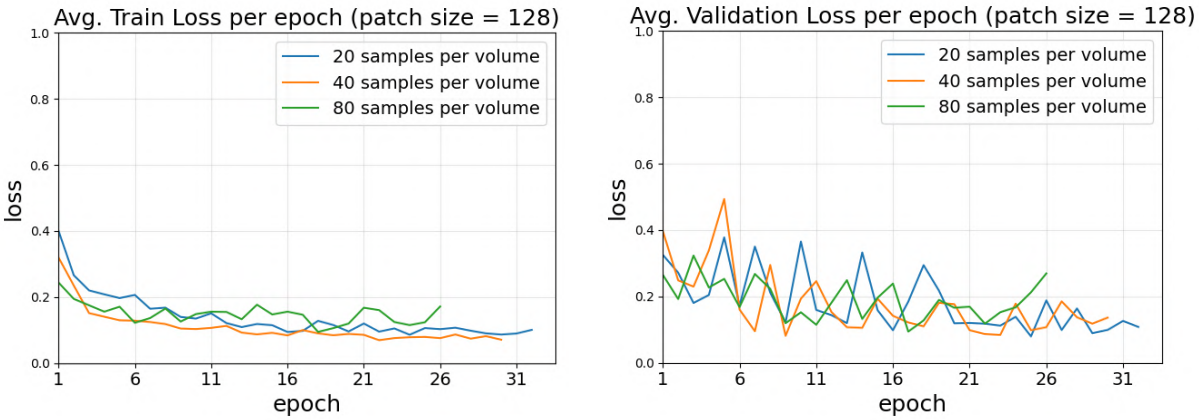


Figure 3.15: Average train and validation losses per epoch for models trained on a patch size of $128 \times 128 \times 128$ voxels, with varying dataset sizes.

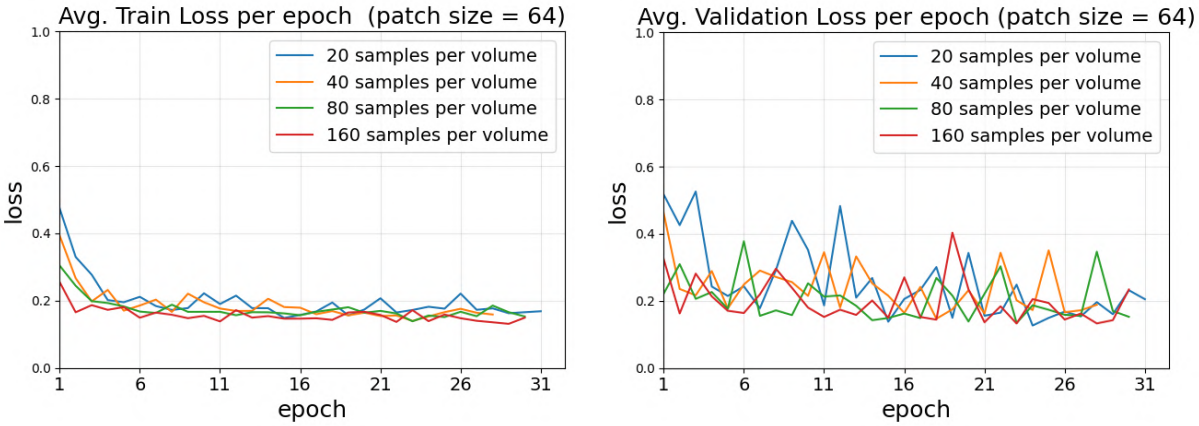


Figure 3.16: Average train and validation losses per epoch for models trained on a patch size of $64 \times 64 \times 64$ voxels, with varying dataset sizes.

Batch size

In line with the dataset size test, a similar experiment was conducted to investigate the optimal batch size. The batch size selection was limited to a maximum of 9 patches for the models trained on patch sizes of $128 \times 128 \times 128$ voxels, due to the limited 80 GB GPU available.

The results that constitute Table 3.3 demonstrate the impact of batch size on GPU memory usage. It is observed that larger batch sizes require more memory, which aligns with expectations. However, the runtimes do not comprise notable differences in the ran experiments. With a focus on optimizing average loss and memory utilization, and taking into account a consistent average test loss across models with different batch sizes, a suitable choice would be a batch size of 9 patches.

Batch Size	Average Test Loss		Runtime		max. GPU (GB)	
	Patch Size		Patch Size		Patch Size	
	128	64	128	64	128	64
3	0.2629	0.2244	10h 50m	7h 59m	41.15	7.58
6	0.3188	0.2427	9h 11m	3h 44m	63.23	9.59
9	0.2504	0.2587	11h 12m	5h 27m	78.80	15.18
20	-	0.2484	-	3h 26m	-	35.18

Table 3.3: Average test loss, runtime, and maximum GPU memory utilized in each experiment for patch dimensions of $128 \times 128 \times 128$ and $64 \times 64 \times 64$ voxels.

As depicted in Figure 3.17 and Figure 3.18, the training and validation processes maintain a consistent behaviour across different batch sizes for the baseline model. For larger batch sizes the convergence is slower requiring a larger number of epochs.

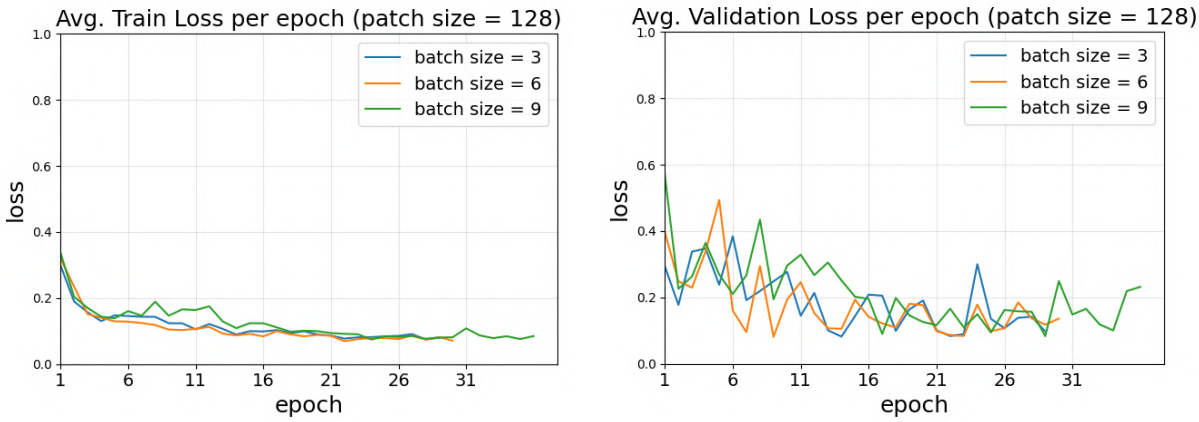


Figure 3.17: Average train and validation losses per epoch for models trained on a patch size of $128 \times 128 \times 128$ voxels, with varying batch sizes.

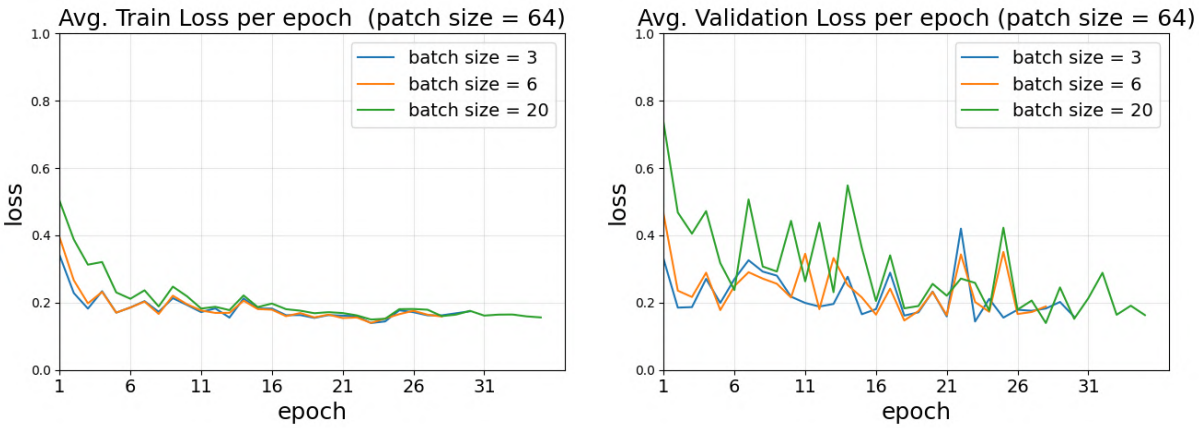


Figure 3.18: Average train and validation losses per epoch for models trained on a patch size of $64 \times 64 \times 64$ voxels, with varying batch sizes.

3.2.2 Final model trained with data augmentation

After discussing the optimal batch size and number of samples per volume in the previous section, it was determined that a batch size of 9 patches and a dataset obtained by randomly selecting 20 patches per volume from the subjects dataset would be used for one last prediction. Considering that 10 transformed subjects were chosen for training and 2 for validation, the training and validation sets consists of 200 and 40 patches, respectively.

The running time required for training and evaluating the models decreased to approximately 5.5 hours for the model trained in patch sizes of $128 \times 128 \times 128$ voxels, and to 3.75 hours for models trained in $64 \times 64 \times 64$ voxel patches. In terms of performance, the average test loss for the first case was 0.1743, whereas for the second case, it was 0.2551. However, it is important to note that these average masked losses were computed based on patch predictions and cannot be directly compared between models trained with different patch sizes.

To conduct a thorough analysis of the results, the entire volume of size $410 \times 410 \times 410$ voxels was re-predicted. This ensures a comprehensive evaluation and allows for meaningful comparisons between different models. From the bar plot depicted in Figure 3.19, it can be appreciated that the greater patch size allows to capture longer and larger axons compared to the model trained with a smaller patch size. Although no significant changes were observed in the prediction from the model trained on patches of $64 \times 64 \times 64$ voxel dimensions, notable differences were appreciated for the larger patch size, compared to the baseline model. As a result, in this subsection, the model trained on the larger patch size will be the main focus of further investigation.

The prominent difference observed in the models trained on a patch size of $128 \times 128 \times 128$ voxels when using smaller training and validation datasets is the exclusion of blood vessels. This is likely due to a limited number of voxels representing this class, which is the minority class, in the fewer randomly selected patches from the subjects in the dataset. In the predicted volume, none of the voxels were classified as blood vessels, despite the ground truth map containing 366,522 voxels labeled as such. The majority of blood vessels were incorrectly labeled as myelin,, with a smaller proportion misclassified as cell clusters and axons (Figure B.6). However, the segmentation of

vacuoles appears to be much more reliable compared to the augmented baseline model, as shown in Figure 3.20.

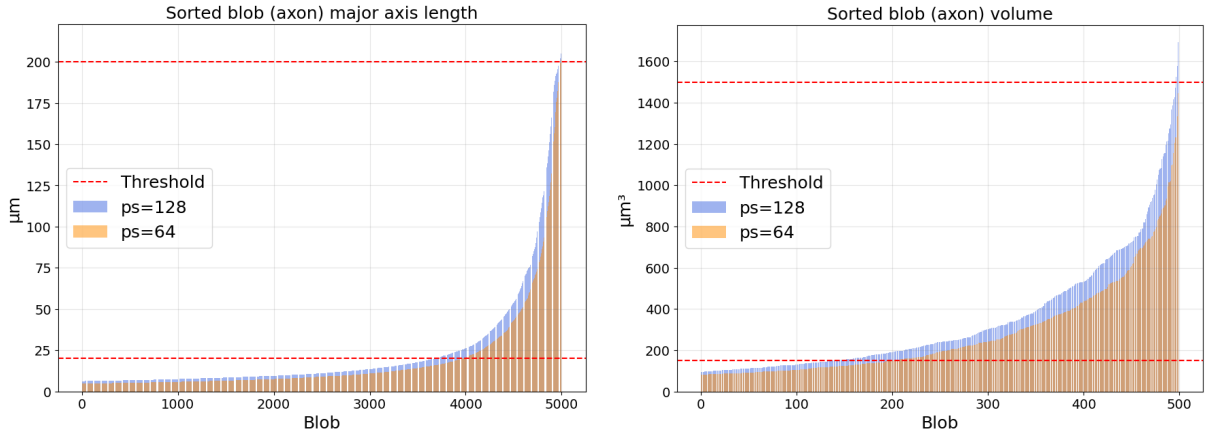
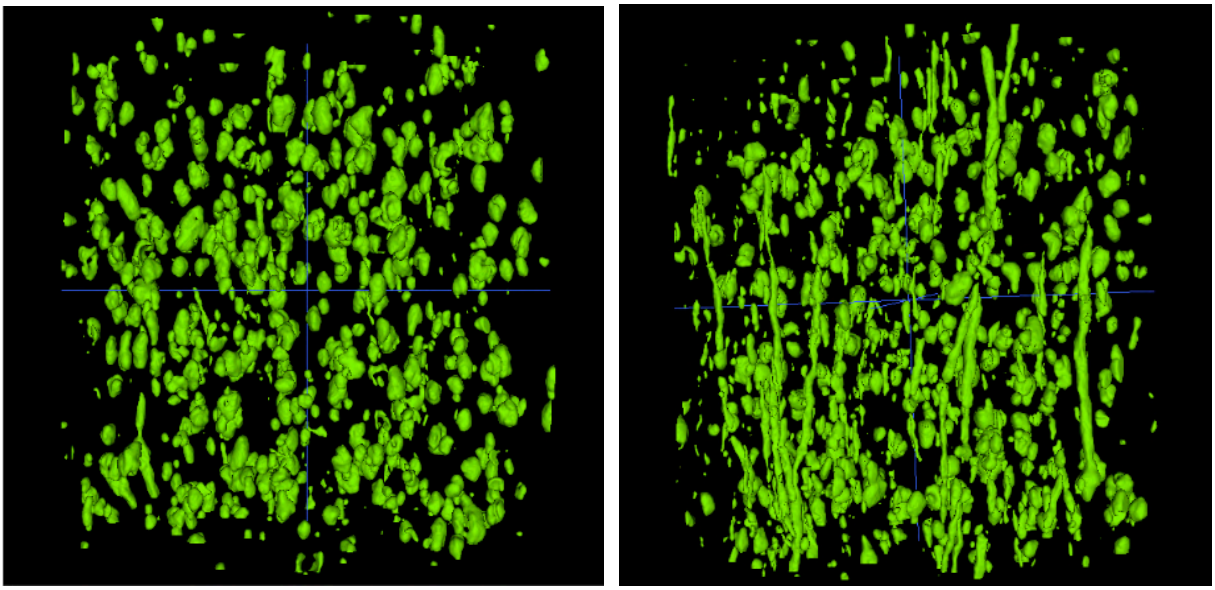


Figure 3.19: Sorted major axis length and volumes of the greatest 5000 and 500 predicted axon segments, respectively.

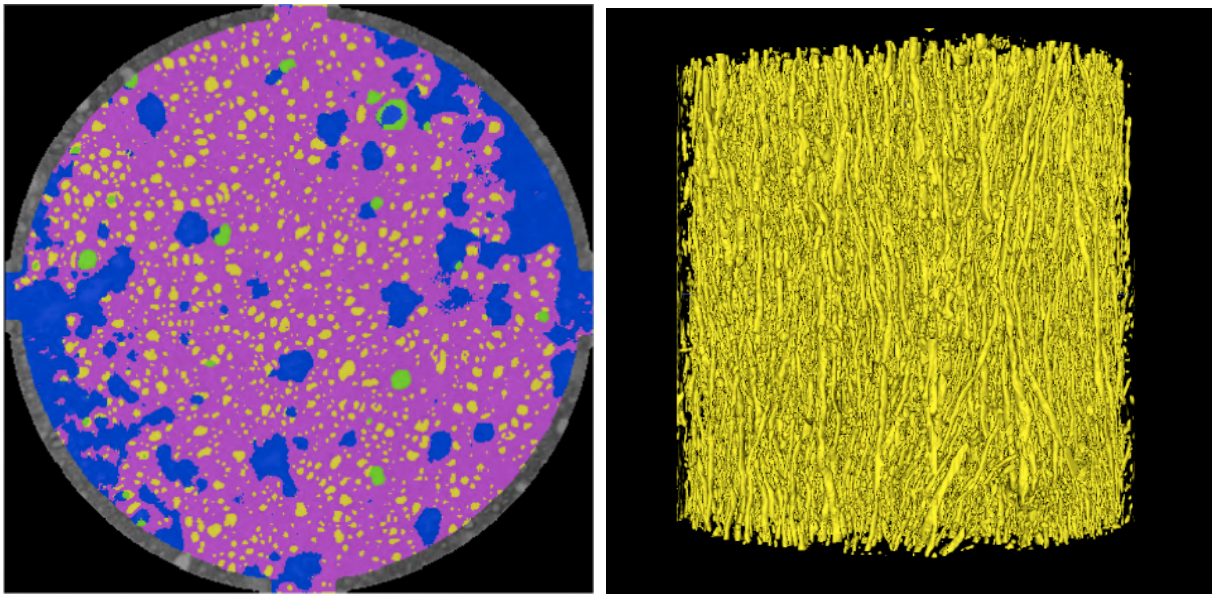


(a) batch size = 9, volumes per sample = 20

(b) batch size = 6, volumes per sample = 40

Figure 3.20: (a) Current model vacuole segmentation. (b) Baseline model vacuole segmentation.

Most importantly, the segmentation of axonal structures is notably more dense for this case, as it can be appreciated in the axial view depicted in Figure 3.21. For a more detailed examination of the 336 individual axon segments, after eliminating artifacts, refer to Figure 3.22. The masked Dice score, calculated with respect to the 54 ground-truth axons, is 0.9521.



(a) **(b)**

Figure 3.21: (a) Axial view of the predicted volume, with axons highlighted in yellow.(b) 3D view of the segmented axonal structures.

Figure 3.22: Axon segment prediction on the initial volume fraction, colored individually.

Finally, the last model was utilized to predict axonal and extra-axonal structures in the entire combined volume of $410 \times 410 \times 1532$ voxel dimensions. The results were satisfactory with 523 axon segments captured after artifact removal. The resulting 3D volume is shown in Figure 3.23, and more closely in Figure B.7. In the overlapping sections that connect different volume fractions, there was a noticeable separation between axonal segments, as shown in Figure 3.24. However, overall, the model effectively addressed this issue and the segments appeared as connected elements in the individual axon labeling map.

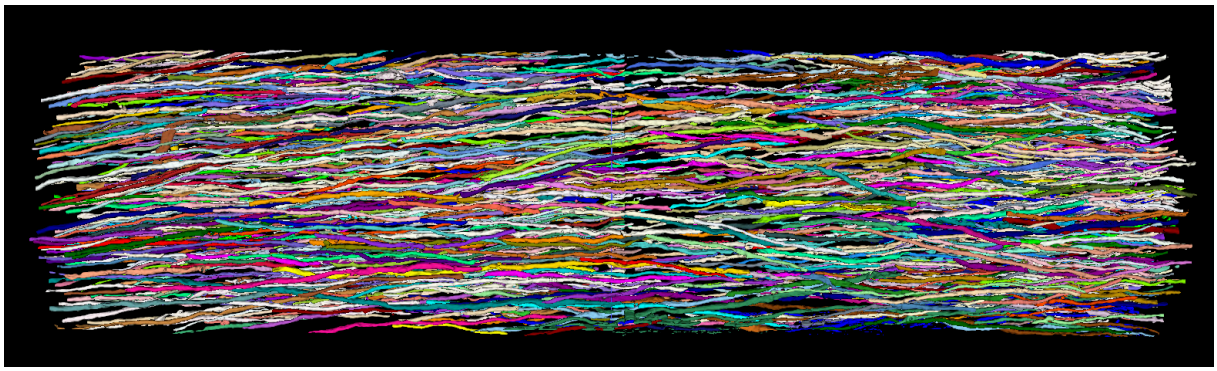


Figure 3.23: Axon visualization the predicted volume with a patch size of $128 \times 128 \times 128$ voxels.

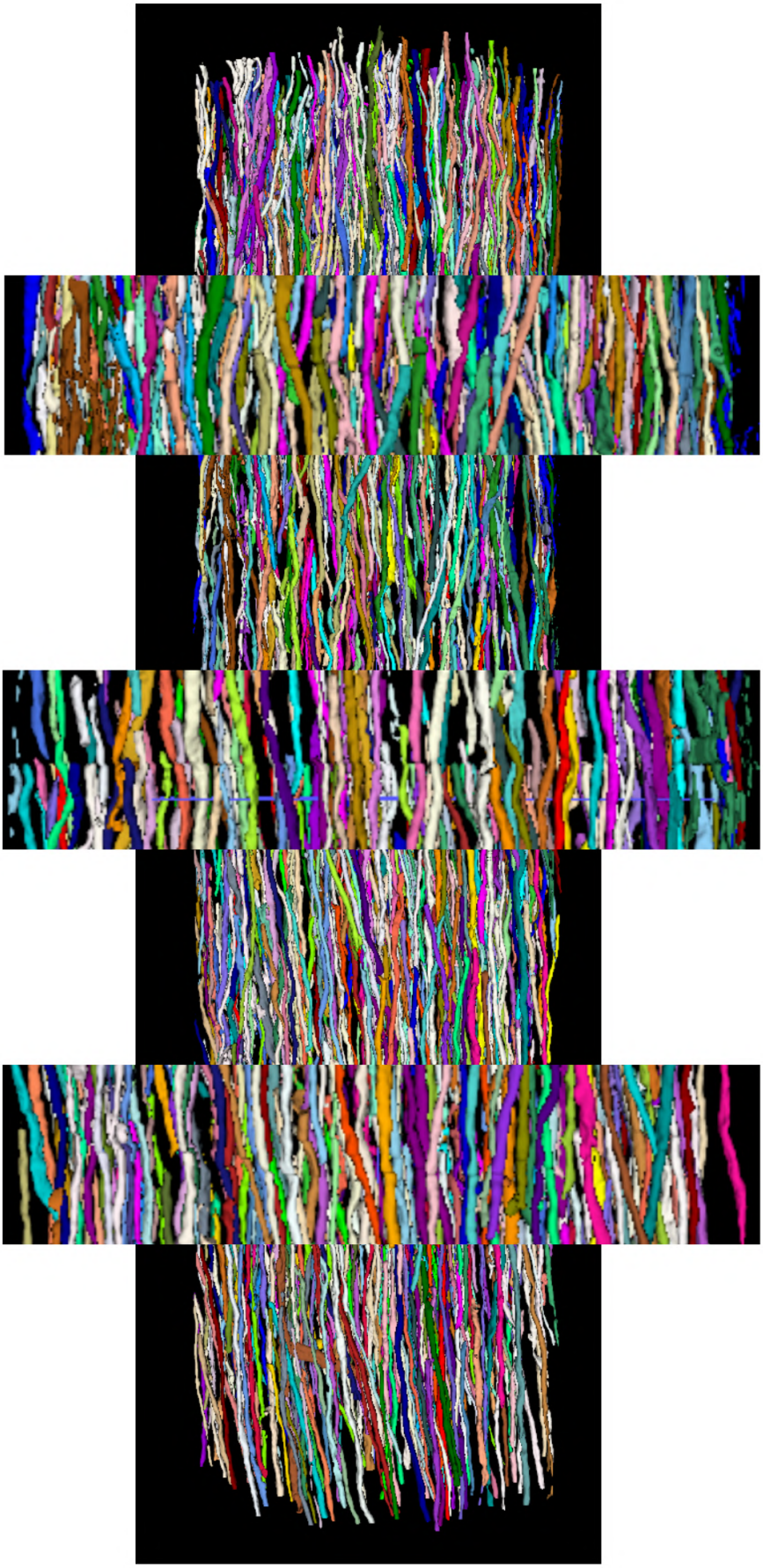


Figure 3.24: Axon separation in the sub-volume overlap.

3.3 Analysis of Cross-hair filter approach

Based on the results presented in Table 3.4, it is evident that a greater kernel does not always yield better outcome. This study's segmentation problem serves as a clear example of this phenomenon, as models trained employing smaller kernel sizes exhibit lower average test losses overall. However, the incorporation of cross-hair filters in the models yields comparable or even superior results in contrast to the models trained using kernels of the original shape.

Model		Average Test Loss	
		Patch Size	
Kernel Size	CH Filter	128	64
3	✗	0.1743	0.2551
3	✓	-	0.3180
5	✗	0.2676	0.3068
5	✓	-	0.2423

Table 3.4: Average test loss for models trained with varying kernel sizes, with and without cross-hair filters. The test loss selected is the masked generalized Dice loss thorough the entire study.

Models trained with a kernel size of $3 \times 3 \times 3$ voxels show minimal performance differences when including cross-hair filters, as expected, since the number of trainable parameters remains the same. However, models trained with a kernel size of $5 \times 5 \times 5$ voxels and incorporating cross-hair filters outperform other models. Nevertheless, there is no clear evidence of outstanding effectiveness of the cross-hair filters. The larger kernel with original shape may potentially contribute to slightly overfitting the model (see Figure 3.25) due to its larger receptive field, which captures more spatial information. Consequently, the increased complexity of the model can dismiss the ability to generalize well to unseen data, resulting in higher test losses.

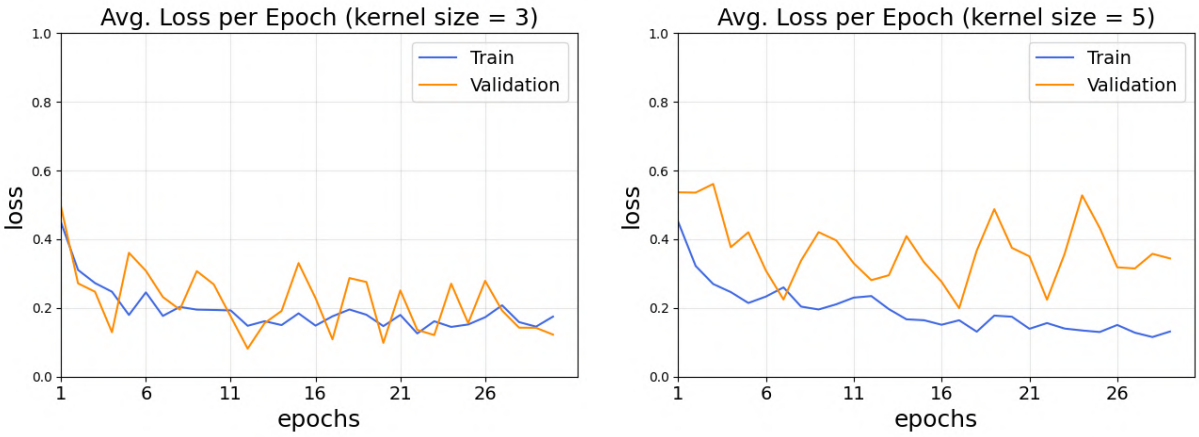


Figure 3.25: Average train and validation losses per epoch for models trained on a patch size of $128 \times 128 \times 128$ voxels, with kernels of dimensions $3 \times 3 \times 3$ and $5 \times 5 \times 5$.

The training process performance of models that include cross-hair filters is similar to models trained with smaller kernel sizes, as it is depicted in Figure 3.26. This comparable behaviour can be attributed to the significantly decreased number of trainable parameters in these models, leading to a simplified model architecture.

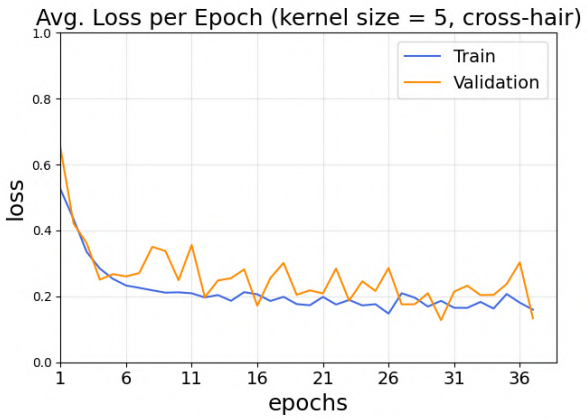


Figure 3.26: Average train and validation losses per epoch for the models trained on a patch size of $64 \times 64 \times 64$ voxels, with a cross-hair kernel of dimension $5 \times 5 \times 5$.

The PyTorch implementation of the cross-hair filters developed in this study, inspired by the TensorFlow implementation by [13], involved three 3D convolutions with 2D filters per each 3D convolution operation in the original network. Although the implementation involved considerably fewer parameters compared to the original shape kernel approach, attempts to optimize time and memory were unsuccessful. This is the case as the main memory usage originates in the intermediate forward activations in this study's implementation. Notably, using cross-hair filters of size 5 and patch size 64 demanded nearly twice the GPU resources compared to conventional convolution filters of identical dimensions (refer to cells marked in red and green, respectively, in Table 3.5). Due to GPU memory constraints, the cross-hair filter could not be tested for kernel size $5 \times 5 \times 5$ and patch size of dimensions $128 \times 128 \times 128$. Furthermore, there were no noticeable runtime advantages observed.

Model		Run Time		max. GPU (GB)	
		Patch Size		Patch Size	
Kernel Size	CH Filter	128	64	128	64
3	✗	5h 30m	3h 47m	78.80	13.28
3	✓	-	1h 58m	-	23.10
5	✗	7h 50m	2h 27m	79.67	14.83
5	✓	-	2h 41m	-	27.86

Table 3.5: Runtime and memory usage for models trained with varying kernel sizes, with and without cross-hair filters.

3.4 Quantification of Diameter Distribution and Length of Segmented Axons

The predictions for the $410 \times 410 \times 410$ voxel volume derived from subsection 3.2.2 were used to quantify different morphological characteristics of axons. To facilitate the extraction of the diameter distribution along axons and their length, a MATLAB script previously employed by M. Andersson et al. in [2] was utilized.

After removing artifacts from the post-processed volume, each axon segment and its corresponding location were extracted based on a connectivity value of 26. A total of 336 unique axons, or axonal segments, were extracted. The centerline of each structure was obtained by calculating a slice-wise centroid estimate from the segmentation, resulting in a single-pixel-wide representation of each component. Small branches (< 7.5 µm) branching off from the main axon branch were discarded.

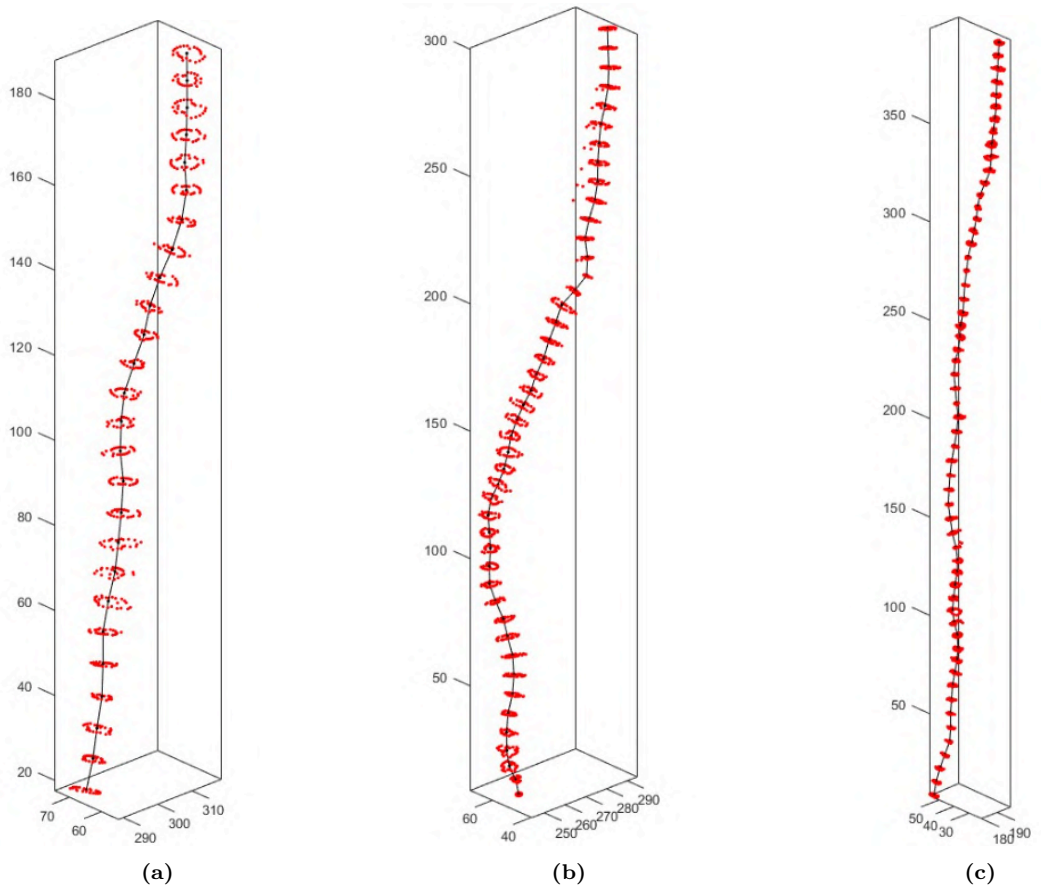


Figure 3.27: Centerline and visualization of considered cross-sectional slices for different predicted axon segments. Axes in pixels.

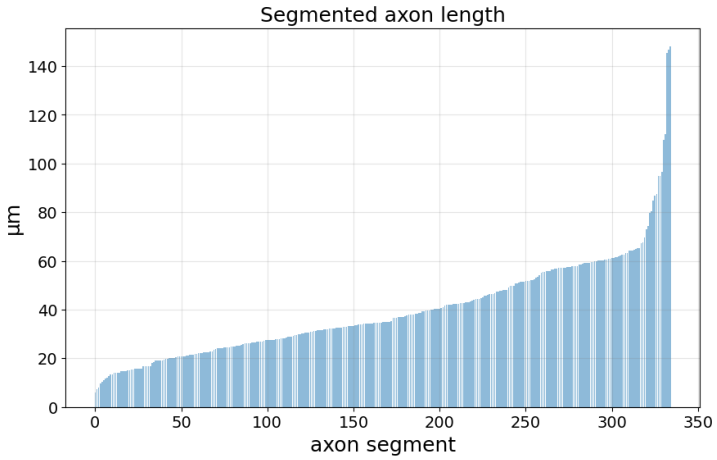


Figure 3.28: Sorted lengths of unique segmented axonal segments.

The resulting 3D centerline curve of each axon segment has two endpoints, representing the start and end points of the respective axon. The axon lengths were determined based on these endpoints, and their distribution is shown in Figure 3.28. Assuming that axons exhibit tubular characteristic, their slice-wise diameter can be approximated as a circular or ellipsoidal shape, as depicted in Figure 3.29.

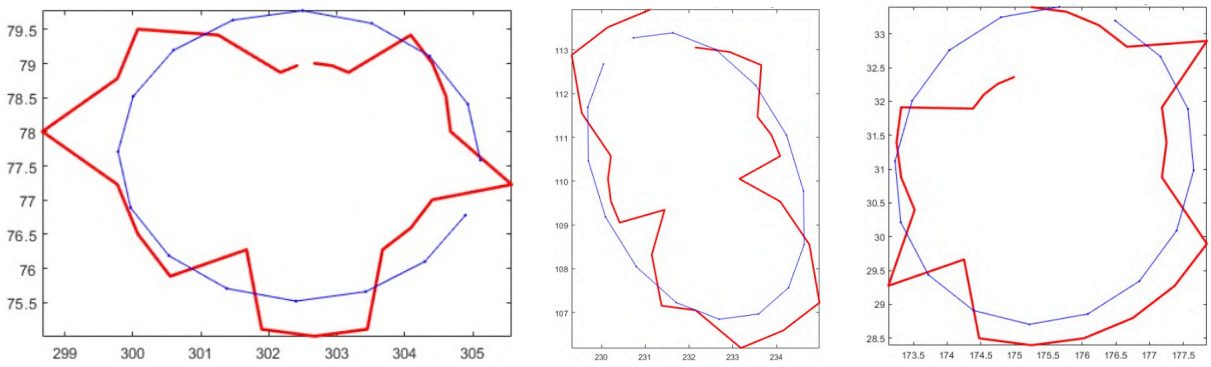


Figure 3.29: Circular and ellipse fit of central cross-sections for various segmented axonal structures.

The cross-sectional area of axons at 7-pixel intervals along each fiber enables to determine the different diameters, resulting in the axon diameter distribution shown in Figure 3.30 for the entire axonal segment population predicted.

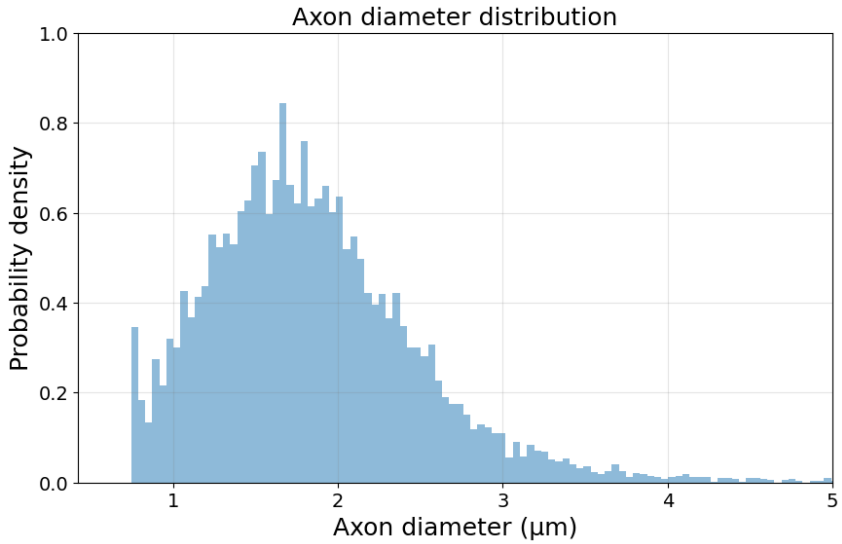
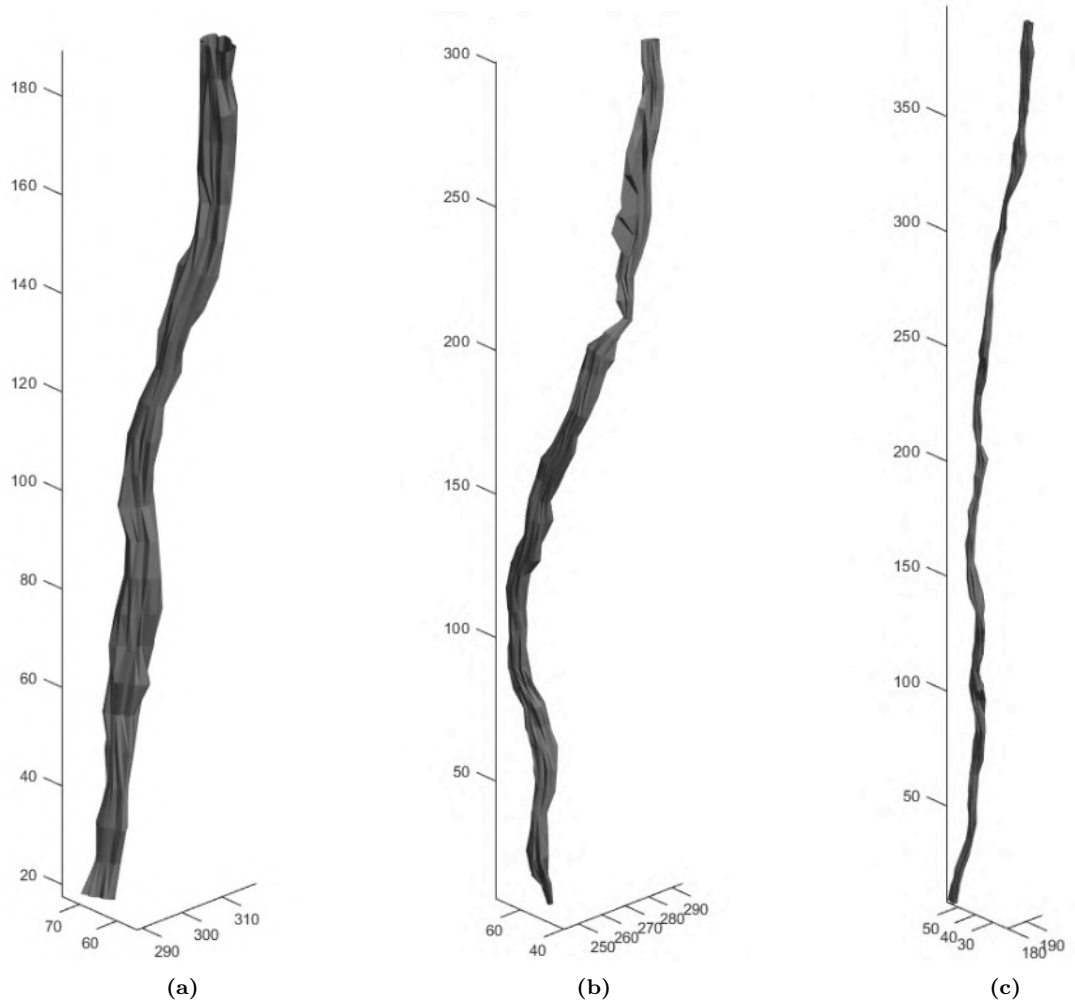


Figure 3.30: Axon diameter distribution consisting of diameter measurements every 7 pixels along all 336 axon segments.

The cross-sectionally sliced mesh for three different axon segments visualized in Figure 3.31 show that the employed model effectively segmented axons with different obliquity, length, diameter and shape.



(a) **(b)** **(c)**

Figure 3.31: Cross sectionally sliced mesh for various segmented axons. Axes in pixels.

Chapter 4

Discussion

In this chapter, the findings from chapter 3 will be summarized by highlighting the key observations and significant outcomes of the study. An overarching interpretation of the research process and results is provided, along with a discussion of the study's limitations.

Throughout the study, U-Net architecture convolutional neural network models were evaluated in parallel using patch sizes of $128 \times 128 \times 128$ and $64 \times 64 \times 64$ voxels. Several loss functions were initially tested for training, and ultimately, the generalized Dice loss was selected due to its remarkable performance. The choice was further enhanced by employing a masking approach, which effectively enabled the model to capture axonal and extra-axonal structures within the unlabeled background of the ground truth. The same loss function was applied for evaluating the model performance during the entire research. The experiment conducted to evaluate the different training loss functions also provided an opportunity to compare the computational resources associated with different patch sizes. As anticipated, larger patch sizes resulted in considerably increased runtime and greater utilization of GPU and RAM memory. However, it is worth noting that the convergence of the models was not directly influenced by the patch size.

The U-Net model trained on the non-augmented original volume, and with smaller patch size, exhibited a slightly sparser annotation, leaving a small number of non-annotated voxels. The length approximation of the predicted axonal segments was comparable for both patch sizes, while the larger patch size revealed slightly larger volume segments. Accordingly, after removing outliers based on major axis length and volume, a total of 313 axon segments were captured for the larger patch size and 300 for the smaller patch size in the combined image volume formed by three individual $410 \times 410 \times 410$ voxel volume fractions. However, the model using a patch size of dimensions $64 \times 64 \times 64$ showed limitations in capturing large elements such as blood vessels — a limitation that was observed throughout the whole dissertation. While no ground truth label map was available for the axon class in the combined volume, the 3D visualization of the prediction provides a clear appreciation of long-tubular structures as axons, as shown in Figure 4.1.

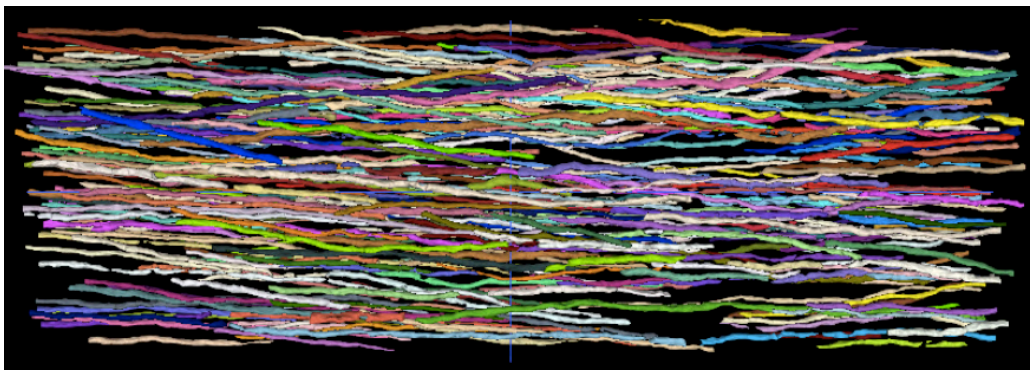


Figure 4.1: Axon visualization the predicted volume discussed in subsection 3.1.2 for a patch size of $64 \times 64 \times 64$ voxel dimensions. Each color indicates an individual axon segment.

After introducing data augmentation, the original volume without any transformation was reserved for testing. This approach enables a more reliable evaluation of performance by utilizing the available ground-truth segmentation within this specific image volume fraction. The training data underwent transformations such as flipping, scaling and rotation, aiming to capture different axon sizes and models that generalize better. When making predictions on the original volume, similar to the non-augmented case, the smaller patch size dismissed blood vessel elements. This is likely because their dimensions do not enable the capture of sufficient shape and voxel intensity information necessary to identify these structures. Although data augmentation results in a denser axon segmentation, the segmentation of other structures in the baseline model generally lacks refinement and, in some cases, is even rougher than in the non-augmented scenario. Moreover, in this case, the model trained with the smaller patch size captured longer and larger axon segments than the model trained in the larger patch size. Consequently, after artifact removal, 191 axon segments were captured with the model trained in a larger patch size, whereas 305 with the smaller patch size. In this case, it could be concluded a better performance of the smaller patch size.

Various dataset sizes and batch sizes were evaluated and analyzed to improve the output while considering time constraints and computational resources. Following this experiment, a decision was made to select a smaller sample size of 200 training patches (compared to the baseline case of 400 training patches) and a slightly larger batch size of 9 patches (compared to 6 in the baseline case). This choice resulted in maximizing GPU resource utilization and reducing the overall runtime, while still achieving satisfactory results based on the masked generalized Dice loss. The resulting model provided a more refined segmentation of axonal and extra-axonal structures for both patch sizes. When using a larger patch size, the model showed a better performance in segmenting larger and longer axons, compared to the model trained with smaller size patches, similar case to the non-augmented output. However, the reduction on the sample size resulted in a failed segmentation of the blood vessel class even for the model trained with a patch size of $128 \times 128 \times 128$ voxels. Nevertheless, the Dice loss restricted to the voxels labeled in the ground truth and to the axon class remained nearly identical for both patch size scenarios.

Figures 4.2 and 4.3 provide a visual comparison of the axon prediction outcomes from the baseline model and the model trained with a batch size of 9 patches and 20 samples per volume. For this comparison, the initial volume fraction was used, after eliminating artifacts in the prediction. As a result, smaller sample size and greater batch size captured considerably more axonal segments for the model trained in patches of $128 \times 128 \times 128$ voxel dimensions, with slightly larger estimated lengths and volumes. For the models trained in a smaller size patches, the results were generally comparable across different models. Thus, an increased patch size can

effectively align with the objective of segmenting a greater number of axons, and varying axon sizes.

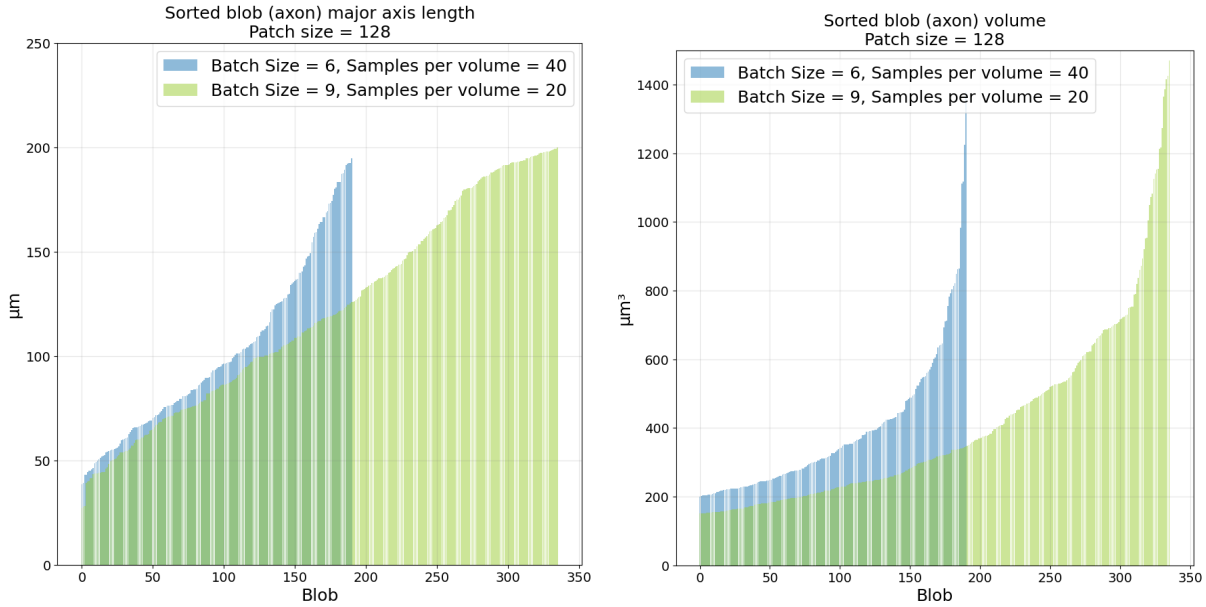


Figure 4.2: Comparison of predicted axon segment measurements for models trained with a patch size of $128 \times 128 \times 128$ voxels.

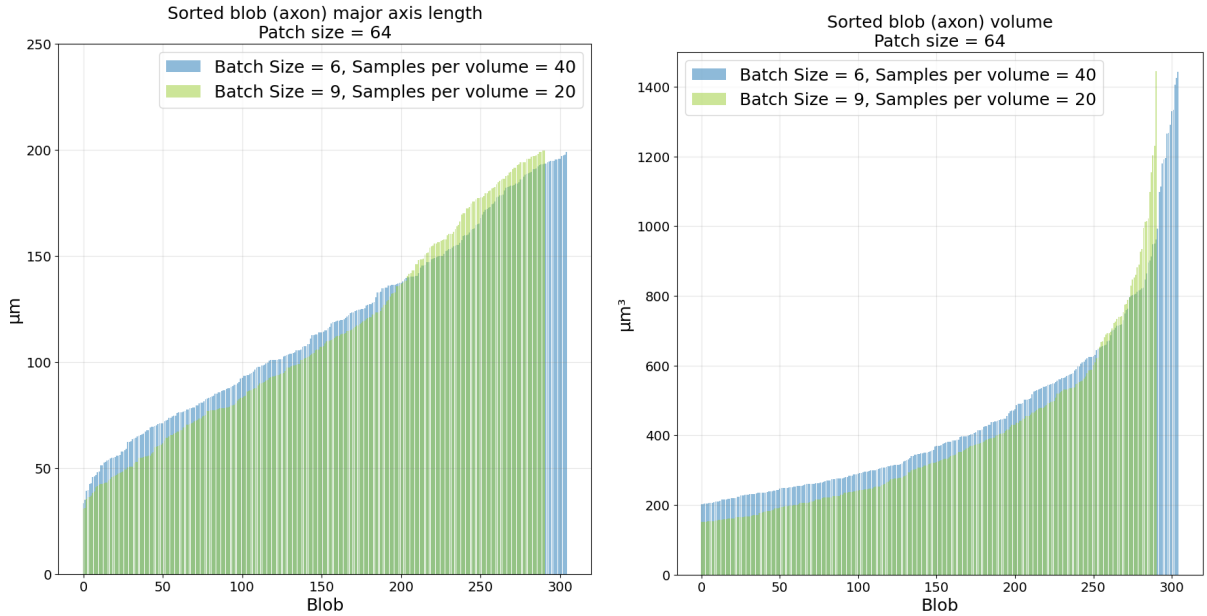


Figure 4.3: Comparison of predicted axon segment measurements for models trained with a patch size of $64 \times 64 \times 64$ voxels.

The lastly explored method was focused in the concept of 'cross-hair' filters, which achieved satisfactory predictions while using a significant fewer number of trainable parameters compared to conventional filters for kernel sizes greater than $3 \times 3 \times 3$ voxels. However, this approach was

not further studied due to the high computational resource demand of the developed implementation in PyTorch. Furthermore, in this evaluation, it was demonstrated that more complex models with a higher number of parameters do not necessarily lead to a improved segmentation of the different axonal and extra-axonal classes.

In line with the original purpose of this dissertation, the focus was narrowed to axons. Accordingly, the final model trained with batch size of 9 patches and a dataset consisting of 20 samples per volume from the set of augmented volumes was selected to generate a conclusive visual representation of the research findings. The prediction results for the complete original image volume available can be observed in Figure 4.4.

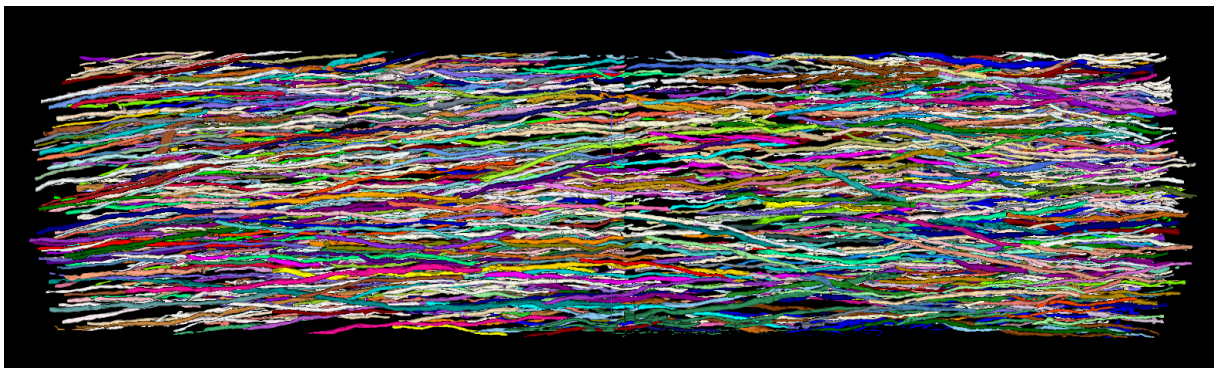


Figure 4.4: Axon visualization the predicted volume with a patch size of $128 \times 128 \times 128$ voxels.

Lastly, the quantification of the diameter distribution along axons and their length, revealed that the study effectively satisfied its purpose of automatically segmenting axons of various sizes based on sparsely annotated and scarce data. The MATLAB script's results indicate that using the major axis length for approximating the length of the segmented axonal structures, as done in previous experiments of this dissertation, tends to overestimate their true length. Additionally, the cross-sectional cuts of axons confirmed their elliptical shape in this view. Furthermore, the segmented structures successfully exhibited lower values in the distribution of axon diameters compared to the ground-truth axons measured by M. Andersson et al. in [2].

In conclusion, a simple model based on a U-Net CNN architecture provides satisfactory results, even for a small dataset. Larger input patch sizes demonstrate better generalization for segmenting axonal and extra-axonal structures. However, when specifically targeting axon segmentation, smaller patch sizes can achieve good results while requiring fewer computational resources. Moreover, the developed model effectively segments small axons (< 2 µm diameter). Incorporating a post-processing step to join axonal segments based on minimum distance and angle variation could further enhance the outcomes of this project. Additionally, a more thorough study of the thousands of unique axonal structures identified in the raw segmentation map may enable the preservation of a greater number of axon segments after artifact removal.

This research faced challenges due to the scarcity of densely annotated data and limited availability of samples for study (a sole segmented image volume). Sparse annotations posed difficulties in the ability to robustly evaluate model performance, resulting in inconclusive and less detailed comparisons in certain cases. Furthermore, the significantly increased computational resources and time required to handle and process 3D image data, as opposed to 2D image data, obstructed the smooth development of this project.

Chapter 5

Conclusions

This thesis aimed to provide a successful segmentation of axonal structures using 3D convolutional neural networks, despite the limited and sparsely annotated data available. The segmentation of extra-axonal structures was also of interest due to their impact on axonal trajectories and potential bias axon diameter measurements. In addition, this research sought to attain a more precise axon diameter estimation compared to diffusion MRI techniques and effectively capture thin axons.

The available volumetric image was extensively analyzed emphasizing the morphological properties of axonal and extra-axonal structures. This enabled informed decision-making regarding the subsequent steps. Various 3D convolutional network architectures were explored, over 2D approaches, targeting to capture spatial context and use correlated information in neighbouring slices. A simple 3D U-Net network architecture model provided effective results matching the aforementioned objectives. However, the introduced 'cross-hair' filter approach was found to be ineffective in the developed implementation.

The output predictions successfully showcased a diverse range of axons with varying characteristics regarding axon diameter, trajectory, and shape. Such findings hold potential to aid in the diagnosis of various medical conditions that specifically target axons of certain sizes.

Notably, in this research study, significantly more microstructural information was extracted from the XNH volumes compared to the initial semi-manual annotations. This would imply a substantial benefit for neurology professionals from the time-saving and increased information provided by the automated extraction process.

Bibliography

[1] Abadi, M., et al. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/

[2] Andersson, M., et al. (2020). Axon morphology is modulated by the local environment and impacts the noninvasive investigation of its structure-function relationship. Proceedings of the National Academy of Sciences, 117(52), 33649-33659. https://www.pnas.org/doi/abs/10.1073/pnas.2012533117.

[3] Andersson, M. (2021). Multi-Modal Microstructural Imaging of Brain White Matter. PhD thesis, Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU Compute).

[4] Cardoso, M. J., Li, W., Brown, R., Ma, N., Kerfoot, E., Wang, Y., Murrey, B., Myronenko, A., Zhao, C., Yang, D., et al. (2022). MONAI: An open-source framework for deep learning in healthcare (Version 1.1.0). arXiv preprint: https://arxiv.org/abs/2211.02701. https://github.com/Project-MONAI/MONAI.

[5] Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., & Ronneberger, O. (2016). 3D U-Net: learning dense volumetric segmentation from sparse annotation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19 (pp. 424-432). Springer.

[6] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint: https://arxiv.org/abs/1412.6980.

[7] Mehus Sunde, B. (2018). Early stopping for PyTorch. Retrieved from https://github.com/Bjarten/early-stopping-pytorch/tree/master

[8] Neptune team. (2019). neptune.ai: Metadata store for MLOps. https://neptune.ai.

[9] Niyas, S., Pawan, S. J., Kumar, M. A., & Rajan, J. (2022). Medical image segmentation with 3D convolutional neural networks: A survey. Neurocomputing, 493, 397-413. Elsevier.

[10] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration (Version 1.13.1). https://pytorch.org.

[11] Pérez-García, F., Sparks, R., & Ourselin, S. (2021). TorchIO: a Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. Computer Methods and Programs in Biomedicine, 106236. https://doi.org/doi:10.1016/j.cmpb.2021.106236.

[12] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18 (pp. 234-241). Springer.

[13] Tetteh, G., Efremov, V., Forkert, N. D., Schneider, M., Kirschke, J., Weber, B., Zimmer, C., Piraud, M., & Menze, B. H. (2020). Deepvesselnet: Vessel segmentation, centerline prediction, and bifurcation detection in 3-d angiographic volumes. Frontiers in Neuroscience, 14, 1285. Frontiers.

[14] Yushkevich, P. A., Piven, J., Hazlett, C., Gimpel Smith, H., Ho, S., Gee, J. C., & Gerig, G. (2006). User-Guided 3D Active Contour Segmentation of Anatomical Structures: Significantly Improved Efficiency and Reliability. Neuroimage, 31(3), 1116-1128.

Appendix A

Output Images of Different Losses in Baseline Model

In this appendix, test examples of each loss compared in subsection 3.1.1 are shown. The baseline model refers to the 3D U-Net network architecture from Figure 2.7 with kernel dimensions $3 \times 3 \times 3$ and no data augmentation.

In each figure the cases of patch sizes of $128 \times 128 \times 128$ and $64 \times 64 \times 64$ voxels are present. The original test patch (image), ground truth segmentation (original mask), predicted segmentation mask (predicted mask), and the performance are compared. The performance visualization has colors: green for correctly classified voxels, red for mislabeled voxels, blue for voxels that were labeled in the ground truth but not labeled in the prediction, and yellow for voxels that were labeled in the prediction but not in the ground truth. The voxels that received a new label (colored in yellow) are not ensured to be correctly classified.

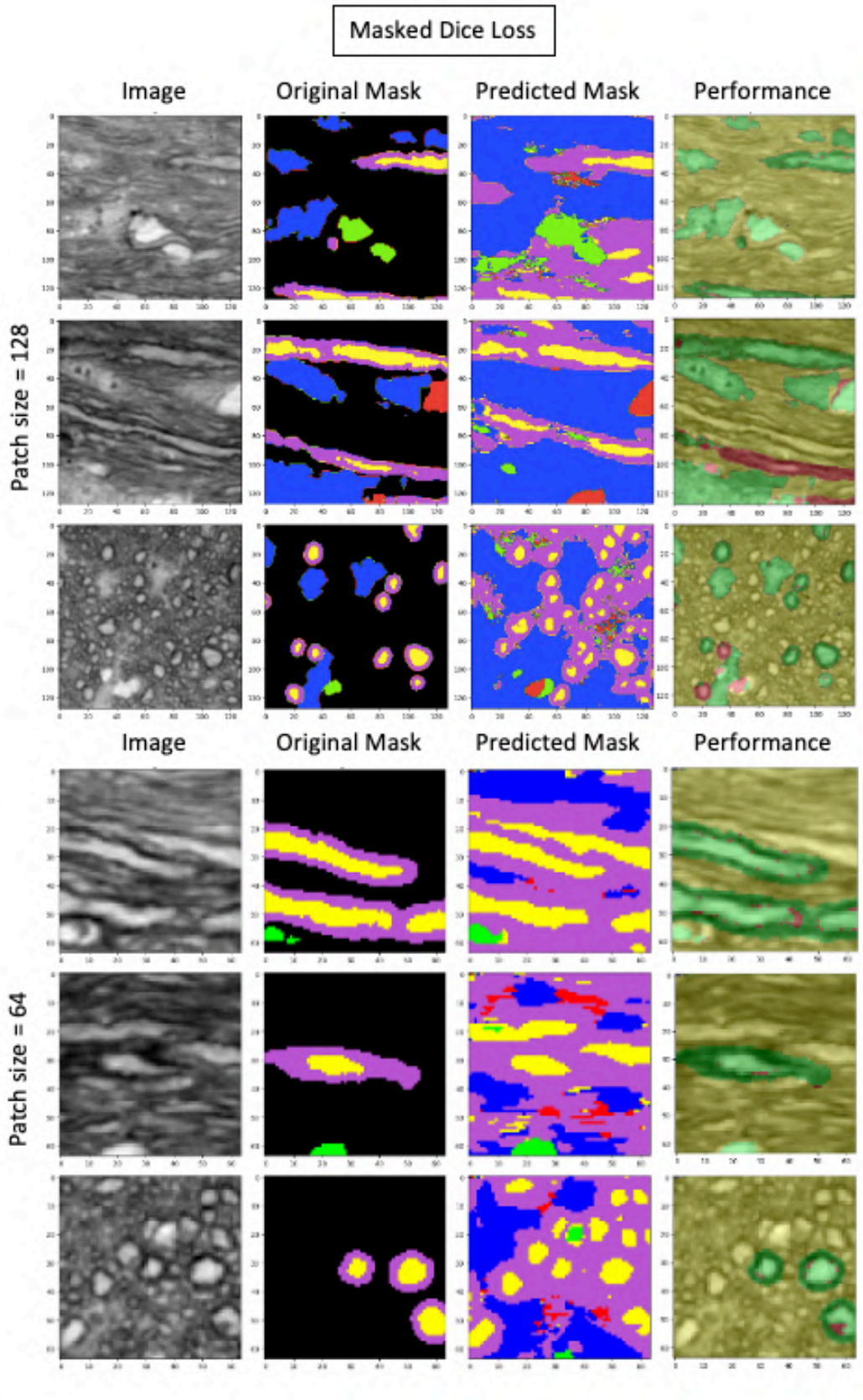

Masked Dice Loss
Image
Original Mask
Predicted Mask
Performance
Patch size = 128
Image
Original Mask
Predicted Mask
Performance
Patch size = 64

Figure A.1

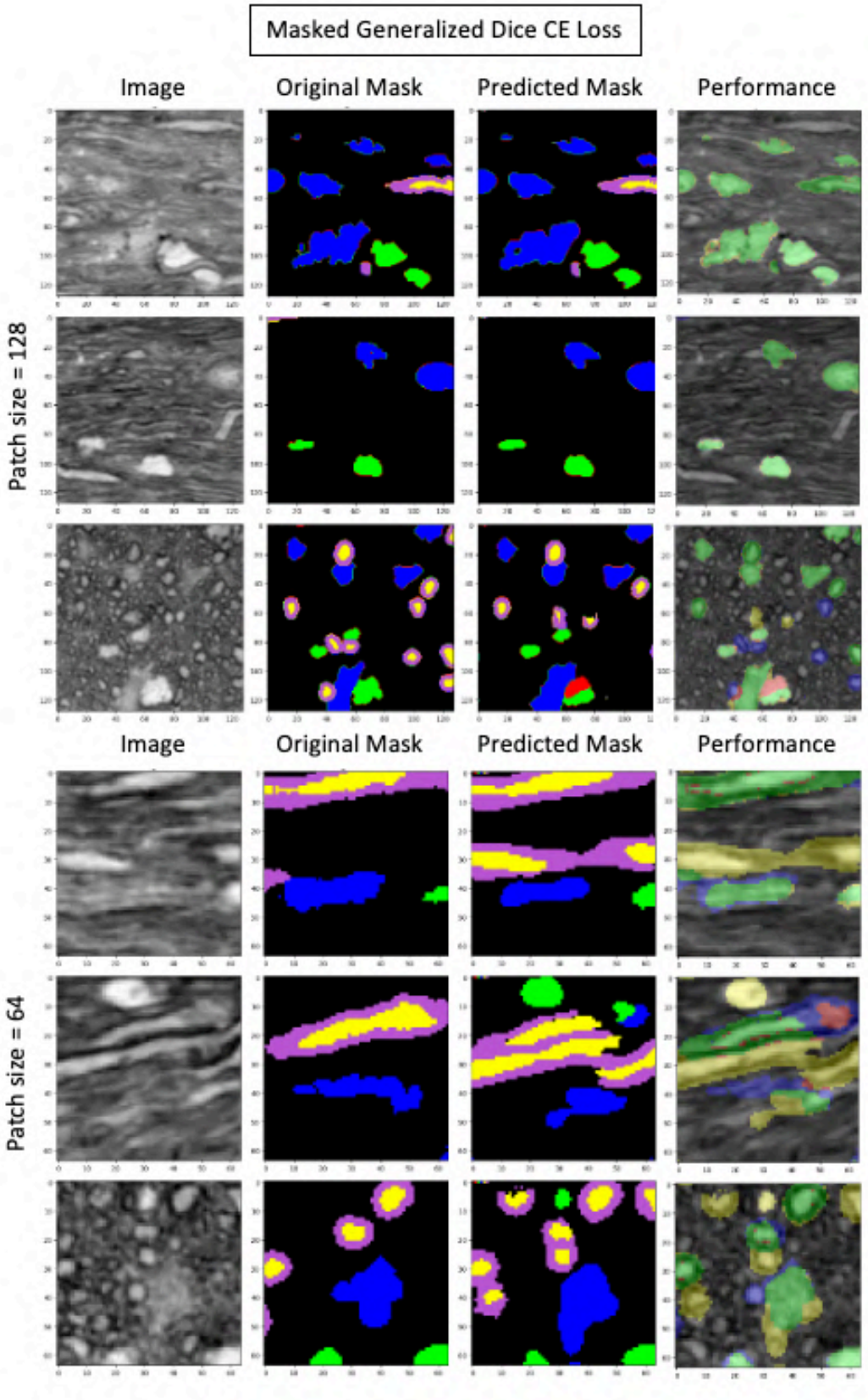

Masked Generalized Dice CE Loss
Image
Original Mask
Predicted Mask
Performance
Patch size = 128
Image
Original Mask
Predicted Mask
Performance
Patch size = 64

Figure A.2

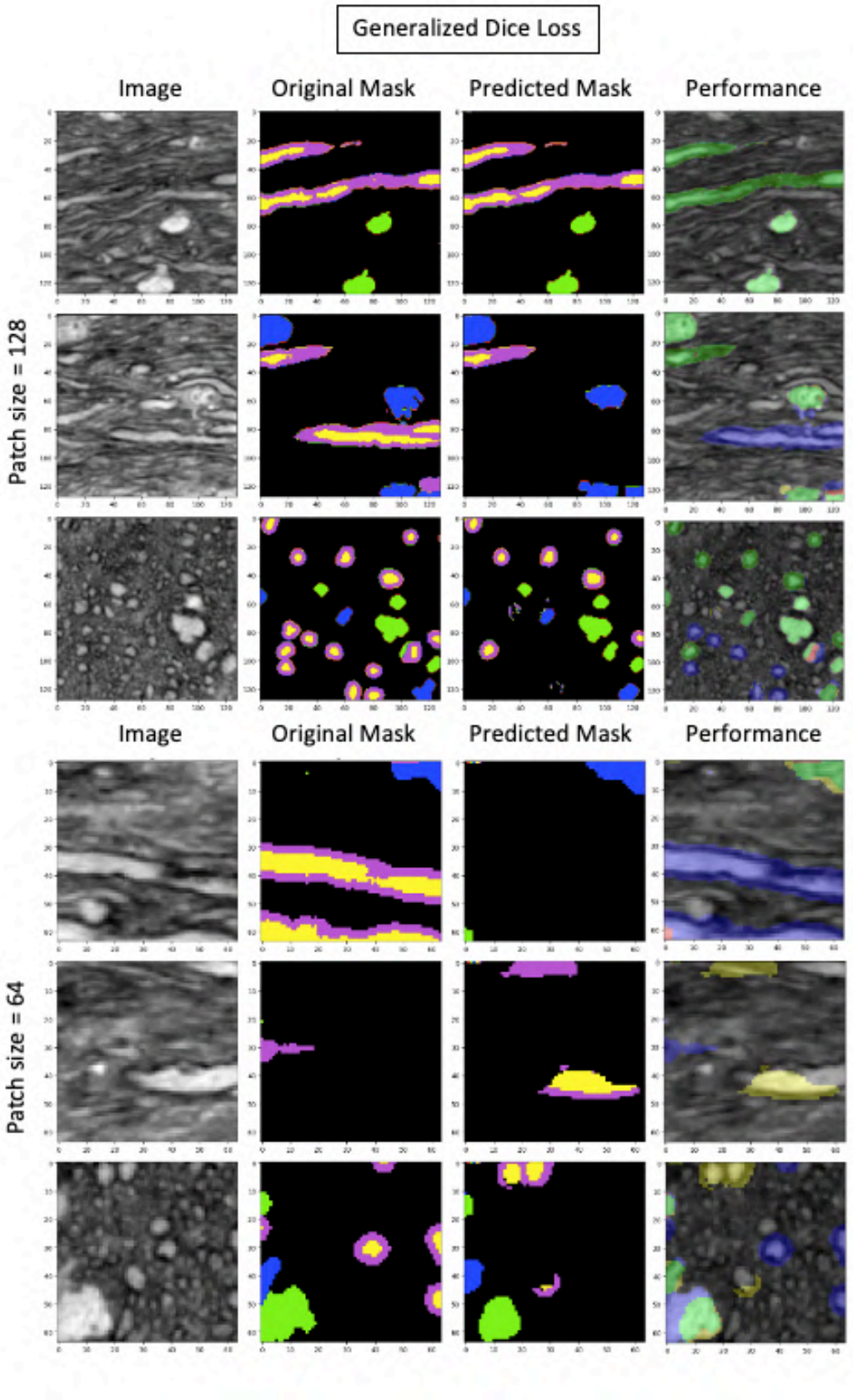

Generalized Dice Loss
Image
Original Mask
Predicted Mask
Performance
Patch size = 128
Image
Original Mask
Predicted Mask
Performance
Patch size = 64

Figure A.3

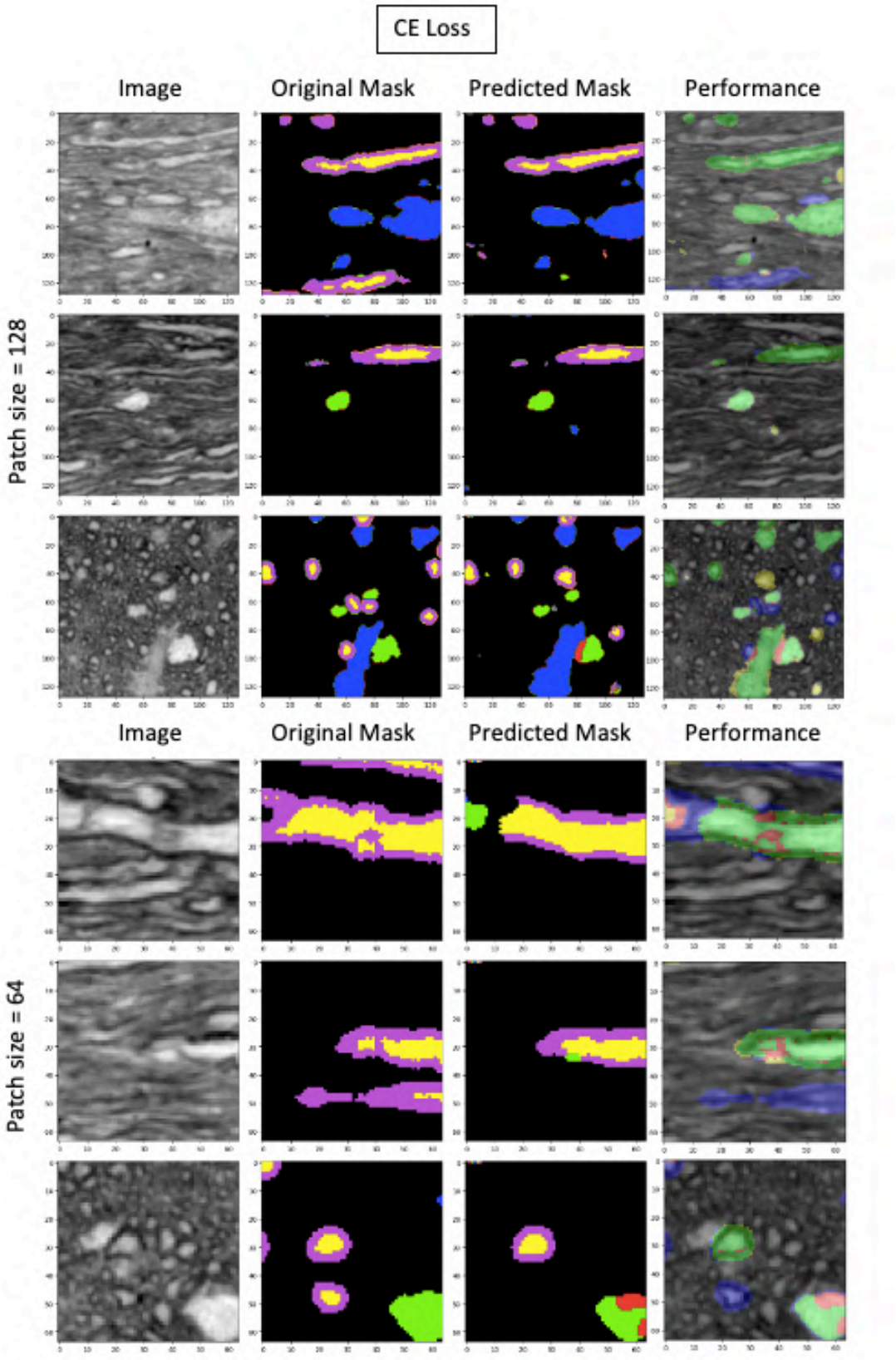

CE Loss
Image
Original Mask
Predicted Mask
Performance
Patch size = 128
Image
Original Mask
Predicted Mask
Performance
Patch size = 64

Figure A.4

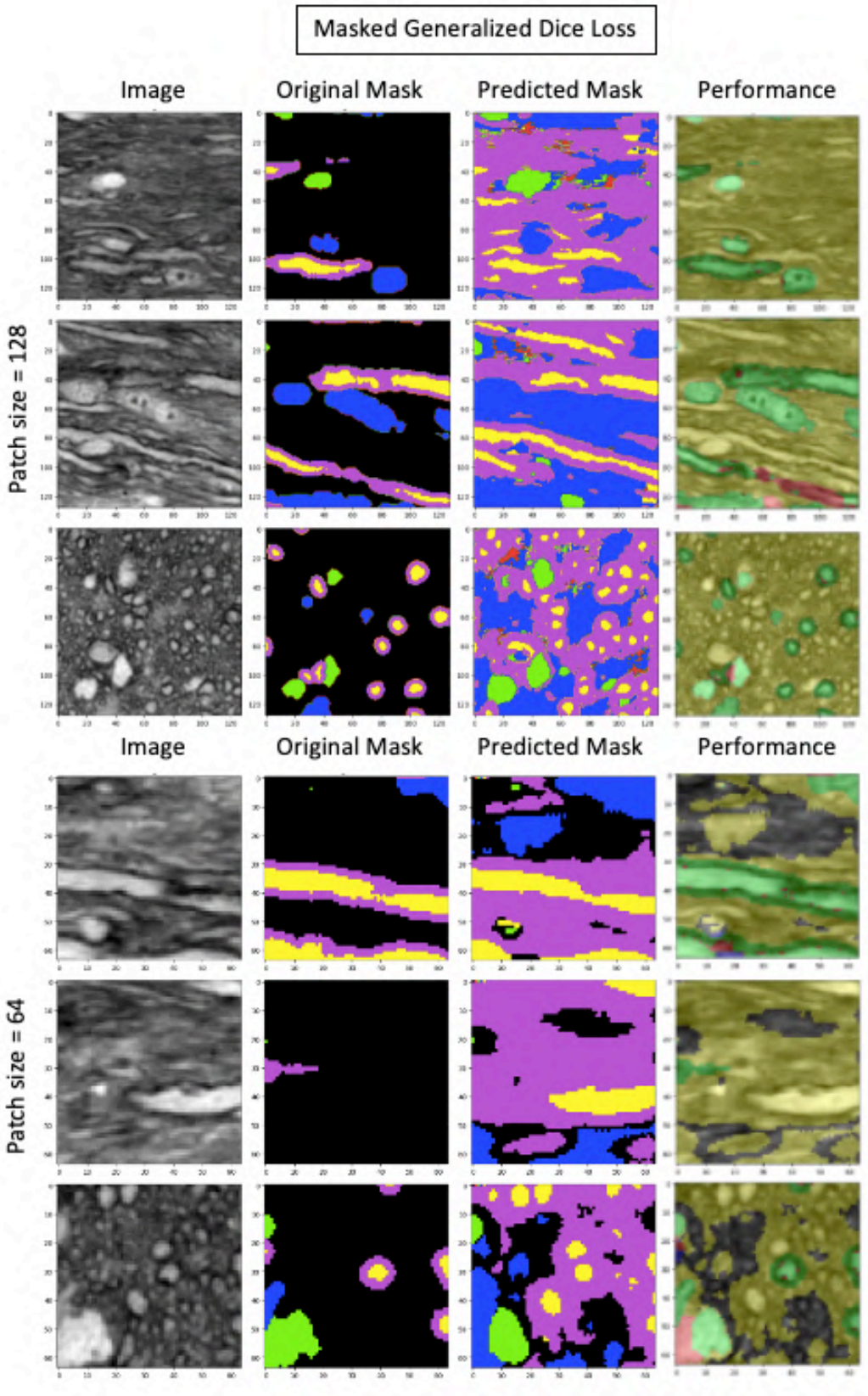

Masked Generalized Dice Loss
Image
Original Mask
Predicted Mask
Performance
Patch size = 128
Image
Original Mask
Predicted Mask
Performance
Patch size = 64

Figure A.5

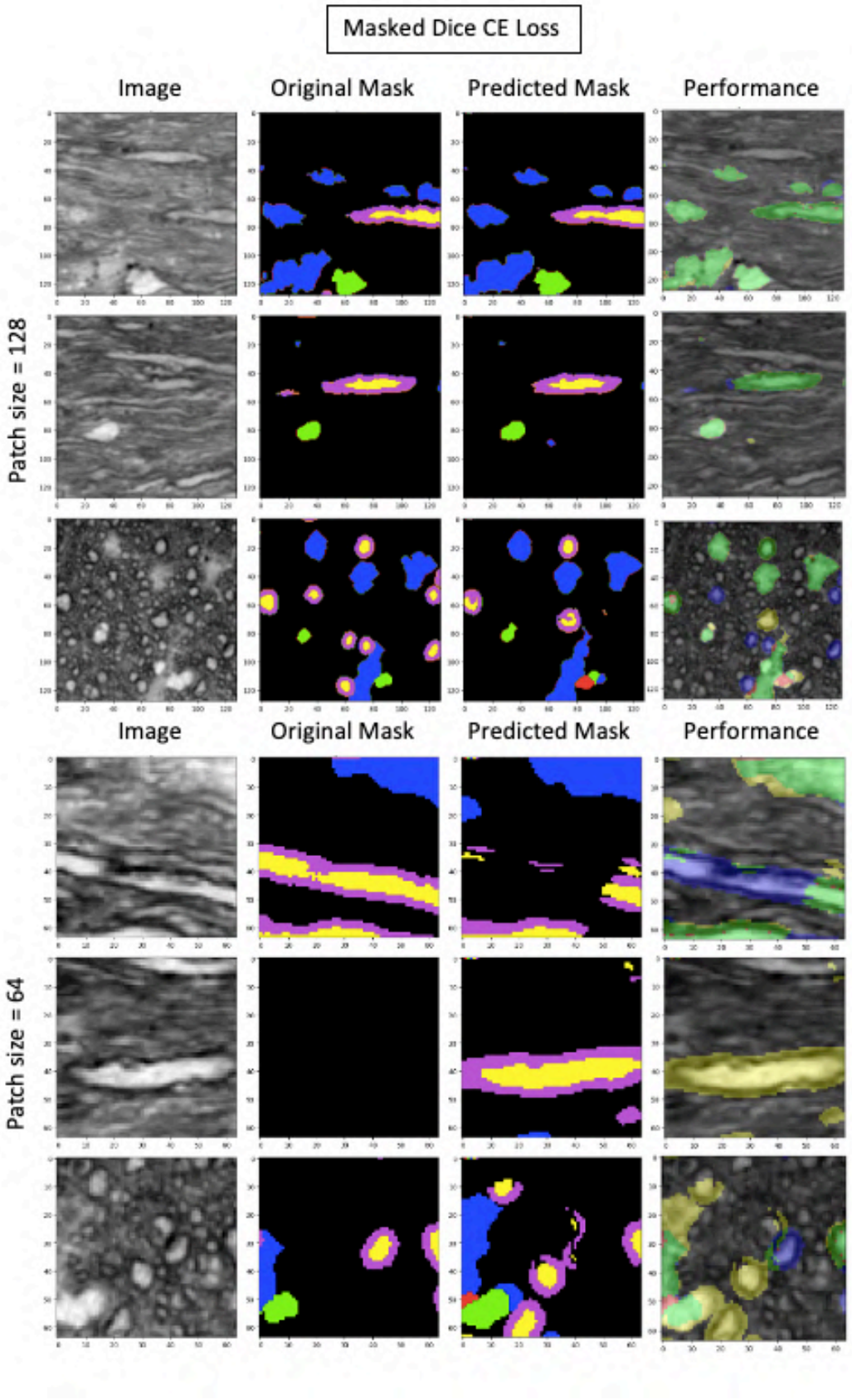

Masked Dice CE Loss
Image
Original Mask
Predicted Mask
Performance
Patch size = 128
Image
Original Mask
Predicted Mask
Performance
Patch size = 64

Figure A.6

Appendix B

Supplementary Images of Axon Segmentation

In this appendix, additional figures are provided to offer further support and visual evidence for the findings commented in chapter 3.

Figure B.1: Axon visualization for the predicted volume from subsection 3.1.2 for a patch size of $128 \times 128 \times 128$ voxels. Each color represents an individual axonal structure.

Figure B.2: Axons in a fraction of the predicted volume from subsection 3.1.2 for a patch size of $128 \times 128 \times 128$ voxels.

Figure B.3: Axon visualization for the predicted volume from subsection 3.1.2 for a patch size of $64 \times 64 \times 64$ voxels. Each color represents an individual axonal structure.

Figure B.4: Axons in a fraction of the predicted volume from subsection 3.1.2 for a patch size of $64 \times 64 \times 64$ voxels.

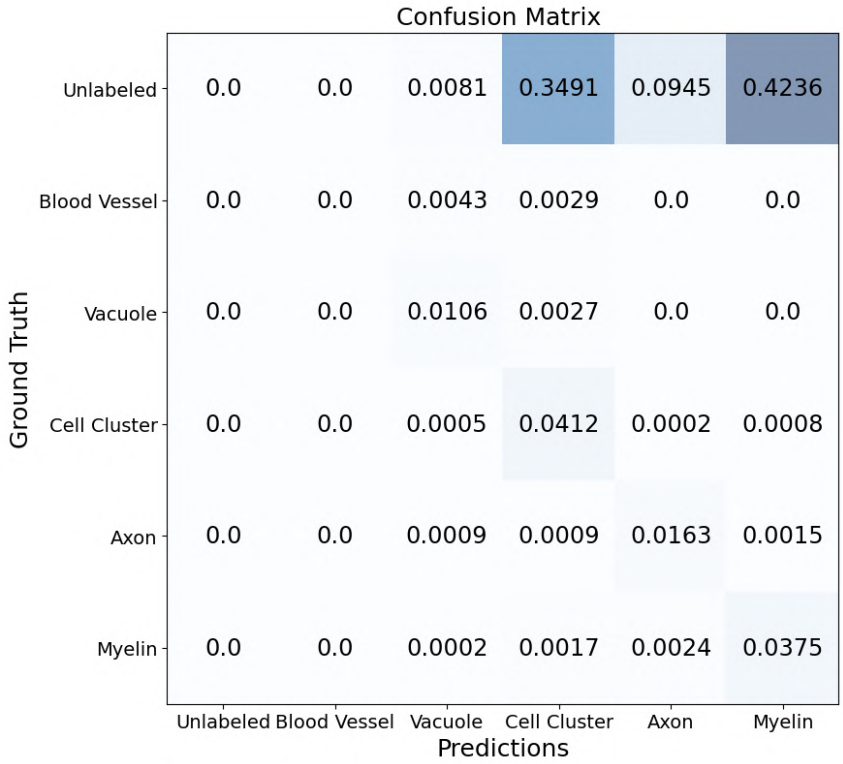


Figure B.5: Confusion matrix depicting the prediction results of the augmented baseline model for a patch size of $64 \times 64 \times 64$ voxels.

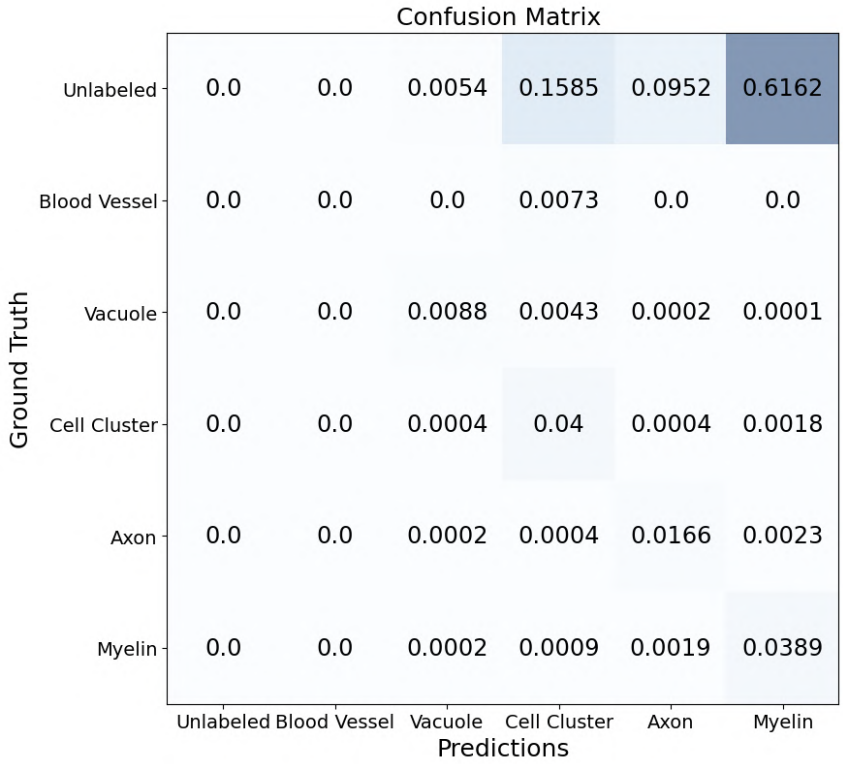


Figure B.6: Confusion matrix illustrating the prediction results of the final augmented model using a patch size of $128 \times 128 \times 128$ voxels, providing evidence for the exclusion of blood vessels in this particular model.

Figure B.7: Axon visualization the predicted volume from subsection 3.2.2 for a patch size of $128 \times 128 \times 128$ voxels. Each color represents an individual axonal structure.

Technical
University of
Denmark

Department of Applied Mathematics and Computer Science
Richard Petersens Plads, Building 324,
2800 Kgs. Lyngby, Denmark
Tlf. +45 4525 3031

compute@compute.dtu.dk
www.compute.dtu.dk