Learning Approach to Delineation of Curvilinear Structures in 2D and 3D Images

Thèse N° 9301

Présentée le 14 juin 2019

à la Faculté informatique et communications
Laboratoire de vision par ordinateur
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Agata Justyna MOSINSKA

Acceptée sur proposition du jury

Prof. B. Merminod, président du jury
Prof. P. Fua, directeur de thèse
Dr C. Becker, rapporteur
Prof. W. Förstner, rapporteur
Prof. F. Schürmann, rapporteur

2019

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Acknowledgements

The last four and a half years were very special for me and I feel that my time at EPFL would not be the same without many people, whom I would like to thank.

First of all, I would like to thank my supervisor, Professor Pascal Fua for giving me a chance to pursue research in his group. Pascal was always very involved in the research process - starting from brainstorming new ideas, to suggesting improvements to our work to publishing the results. I would also like to thank my Thesis Jury Committee, Doctor Carlos Becker, Professor Wolfgang Förstner, Professor Bertrand Merminod and Professor Felix Schürmann for agreeing to review this thesis.

The beginning of any endeavor can be challenging but thanks to Raphael Sznitman I had a lot of support at the start of my PhD. I really appreciate the time spent during our weekly meetings, where I could share with him my progress and any difficulties. Furthermore, I would also like to thank Przemek Glowacki for introducing me to the topic of my thesis, his help with the code and of course our coffee breaks. Just like the beginnings, also the endings may be quite demanding. However, thanks to Mateusz Kozinski the finish of my PhD was both productive *and* fun time.

I owe my special thanks to our secretaries - Josiane and Ariane - for always being on top of things, ready to answer any questions I had and arranging all travel and work-related issues. I would also like to thank them for giving me a lot of useful advice about the life in Switzerland.

Throughout my time at EPFL I shared the office with Dat, Amos, Roger and Semih, who were great companions on this journey and shared with me many ups and downs.

During those four years, I met extremely talented and friendly colleagues, with whom I spent quality time both at work and outside of it. I would like thank Helge, for sharing many happy (and sometimes scary) moments during our outdoor adventures. My another great companion in the beginning of my PhD was Yannick, always full of positive energy and ideas for hikes. I would also like to thank Amaury and Isinsu for our countless discussions, work-outs and dinners that always brought smile on my face. My time at CVLab also would not be the same without Stefano, Pablo Eduard and Sena.

The first year of my PhD was particularly enjoyable thanks to my fellow EDIC students — Handan, Utku and Elie.

When I was coming to Lausanne I could not have wished for better friends than those that I

met. Thank you Onur and Nathalie — for your compassion, kind words and our trips. Dora and Andreas — for your understanding, delicious dinners and climbing. Manos — for showing me a different point of view, at the same time being very open-minded. Anli — for many wine tastings and weekend trips. Agata and Maria — for radiating your positive energy on my everyday life.

I would not be in this place without my parents. They always offered me their support and encouragement, often sacrificing their own interests. I hope that today they feel it was worth it.

Lastly, I would like to thank my friend, my husband and my rock — Konrad. For his love and believing in me, pushing me out of my conmfort zone (with sometimes different consequences), making me laugh even in the hardest times and sharing with me every day of this PhD.

Lausanne, December 10, 2018 A. M.

Abstract

Detection of curvilinear structures has long been of interest due to its wide range of applications. Large amounts of imaging data could be readily used in many fields, but it is practically not possible to analyze them manually. Hence, the need for automated delineation approaches. In the recent years Computer Vision witnessed a paradigm shift from mathematical modelling to data-driven methods based on Machine Learning. This led to improvements in performance and robustness of the detection algorithms. Nonetheless, most Machine Learning methods are general-purpose and they do not exploit the specificity of the delineation problem. In this thesis, we present learning methods suited for this task and we apply them to various kinds of microscopic and natural images, proving the general applicability of the presented solutions.

First, we introduce a *topology loss* - a new training loss term, which captures higher-level features of curvilinear networks such as smoothness, connectivity and continuity. This is in contrast to most Deep Learning segmentation methods that do not take into account the geometry of the resulting prediction. In order to compute the new loss term, we extract topology features of prediction and ground-truth using a pre-trained network, whose filters are activated by structures at different scales and orientations. We show that this approach yields better results in terms of conventional segmentation metrics and overall topology of the resulting delineation.

Although segmentation of curvilinear structures provides useful information, it is not always sufficient. In many cases, such as neuroscience and cartography, it is crucial to estimate the network connectivity. In order to find the graph representation of the structure depicted in the image, we propose an approach for joint segmentation and connection classification. Apart from pixel probabilities, this approach also returns the likelihood of a proposed path being a part of the reconstructed network. We show that segmentation and path classification are closely related tasks and can benefit from the synergy.

The aforementioned methods rely on Machine Learning, which requires significant amounts of *annotated* ground-truth data to train models. The labelling process often requires expertise, it is costly and tiresome. To alleviate this problem, we introduce an Active Learning method that significantly decreases the time spent on annotating images. It queries the annotator only about the most informative examples, in this case the hypothetical paths belonging to the structure of interest. Contrary to conventional Active Learning methods, our approach exploits

local consistency of linear paths to pick the ones that stand out from their neighborhood. Our final contribution is a method suited for both Active Learning and proofreading the result, which often requires more time than the automated delineation itself. It investigates edges of the delineation graph and determines the ones that are especially significant for the global reconstruction by perturbing their weights. Our Active Learning and proofreading strategies are combined with a new efficient formulation of an optimal subgraph computation and reduce the annotation effort by up to 80%.

Keywords: COMPUTER VISION, Machine Learning, curvilinear structures, segmentation, Active Learning.

Zusammenfassung

Die automatisierte Erkennung von kurvilinearen Strukturen ist aufgrund ihres breiten Anwendungsspektrums seit langem von Interesse. Große Datenmengen werden in vielen Bereichen verwendet, eine manuelle Analyse ist jedoch noch nicht praxistauglich. Daher besteht ein großer Bedarf an automatisierter Delineation. Computer Vision erlebte einen Paradigmenwechsel von der mathematischen Modellierung hin zum datengetriebenen Ansätzen und Maschinellem Lernen. Dies führte zu Verbesserungen der Leistung und Robustheit der Erkennungsalgorithmen. Nichtsdestotrotz sind die meisten Methoden zu generell und können daher die Besonderheit der Delineation nicht ausnutzen. In dieser Arbeit präsentiere ich speziell für diese Aufgabe ausgerichtete Lernmethoden und zeige ihre allgemeine Anwendbarkeit an Hand von verschiedenen Arten von mikroskopischen und natürlichen Bildern.

Zunächst wird einen *Topology Loss* eingeführt - eine neue Metrik für überwachtes Lernen, welcher übergeordnete Merkmale krummliniger Neuronalen Netzwerk wie Glätte, Konnektivität und Kontinuität erfasst. Dies steht im Gegensatz zu den meisten existierendenn Deep Learning-Segmentierungsmethoden, bei denen die Form der Ausgabe nicht berücksichtigt wird. Um die neue Metrik zu berechnen, extrahieren wir Topologiemerkmale der Ausgabe und des Referenzbildes mit einem Netzwerks. Diese Netzwerk Filter erkennen Strukturen mit unterschiedlichen Maßstäben und Orientierungen. Dieser Ansatz liefert bessere Ergebnisse in Bezug auf herkömmliche Segmentierungsmetriken und die Gesamttopologie der Segmentierung.

Eine Segmentierung liefert zwar nützliche Informationen über die Struktur, ist jedoch nicht immer ausreichend. In vielen Fällen is es von entscheidender Bedeutung, die Netzwerkverbindungen abzuschätzen. Um die Netzwerktopographie der im Bild dargestellten zu erhalten, schlagen wir einen Ansatz für die gemeinsame Segmentierung und Verbindungsklassifizierung vor. Abgesehen von Pixelwahrscheinlichkeiten gibt dieser Ansatz auch die Wahrscheinlichkeit zurück, mit welcher ein bestimmter vorgeschlagener Pfad ein Teil des Berechneten Netzwerks ist. Wir zeigen, dass diese beiden Aufgaben eng miteinander verbunden sind und der Algorithmus von deren Synergien profitiert.

Die beschriebenen Methoden basieren auf maschinellem Lernen, was bedeutende Mengen an *annotierten* Trainingsdaten erfordert. Der Annotationsprozess erfordert häufig Fachwissen, ist kostspielig und zeitraubend. Um dieses Problem zu lösen, führen wir eine Active-Learning-

Methode ein, die den Zeitaufwand für das Annotieren von Bildern erheblich verringert, indem der Annotator nur nach den informationshaltigsten Beispielen, in diesem Fall den hypothetischen Pfaden, die zur interessierenden Struktur gehören, abgefragt wird. Im Gegensatz zu herkömmlichen Methoden des Aktiven Lernens nutzt unser Ansatz die lokale Konsistenz linearer Pfade, um diejenigen auszuwählen, welche sich aus ihrer Nachbarschaft hervorheben. Unser letzter Beitrag ist eine Methode, die sowohl für Aktives Lernen als auch für das Korrekturlesen des Ergebnisses geeignet ist. Teile der Delineation werden untersucht und diejenigen ermittelt, die für die globale Rekonstruktion von besonderer Bedeutung sind. In Kombination mit einer neuen Formulierung optimaler Subgraphberechnung reduzieren die vorgeschlagenen Strategien den Annotierungsaufwand um 80%.

Schlüsselwörter: Computer Vision, Machine Learning, kurvilinearen Strukturen, Segmentierung, Active Learning

Contents

List of Figures

List of Tables

1 Introduction

Curvilinear structures are prevalent in nature, appearing on a wide range of scales: from neurons and blood vessels in biomedical images to cracks in materials and roads in satellite images, some of which are shown in Fig. 1.1. Their detection has been an important task in Computer Vision since the dawn of the field, which should not be surprising given numerous applications such as neuron reconstruction and road network extraction. However, after 40 years of sustained research efforts, the problem remains unsolved, mostly due to the large variety of appearances and sizes of such structures. At the same time, advances in imaging technologies allowed obtaining unprecedented amounts of high quality data, that can be used in a large variety of applications. However, their manual analysis is tedious and time consuming, leading to a lot of valuable data remaining unused. To give an example, the total size of road network in the USA alone exceeds 6.5 million killometers and it is quickly evolving in countries such as China or India. Furthermore, neuroscientists estimate that there are around 100 billion neurons in an average human brain, but manual annotation of microscopic image depicting one of them may take up to several days. There is therefore a great need for efficient and robust automated delineation methods that can quickly provide reliable reconstructions of curvilinear structures and which can be used for further analysis.

In recent years, Computer Vision has seen a paradigm shift from mathematical modelling to learning-based methods. In contrast to traditional methods, Machine Learning and, even more recently, Deep Learning, makes use of the vast amounts of available data by finding common, informative patterns. It has been successfully applied to many vision problems including image classification [53], object detection [59, 87] and semantic segmentation [52, 88, 14].

However, it appears that conventional general-purpose Machine Learning techniques are often not enough when applied to delineation. This is because certain characteristics of curvilinear structures, such as smoothness, orientation and continuity, are simply ignored when designing a new algorithm. In this thesis, we show that adapting general-purpose learning methods

by exploiting structural information can considerably improve delineation results and help avoid some of the common pitfalls of the traditional methods. We also describe how it can considerably reduce annotation effort involved in obtaining ground-truth data essential for training models, which usually is costly to obtain and involves expert knowledge. In short, our methods enable fast analysis of various kinds of images depicting curvilinear structures with minimal user interaction.

The problem presented in this thesis can be described as follows: given a 2D image or 3D image stack, detect curvilinear structures and find their underlying graph representation with minimum user interaction (including annotating training data and proofreading the resulting reconstruction). The work presented in this thesis tackles the essential steps of delineation algorithm, such as performing an initial segmentation, inferring the connectivity of the detected segments, as well as obtaining ground-truth data for training and proofreading the result.

The remainder of this chapter is structured as follows: we first discuss applications, which can benefit from automating the delineation process. Subsequently, we name the key challenges that one faces working with curvilinear structures and finally we present our contributions that tackle the aforementioned problems.

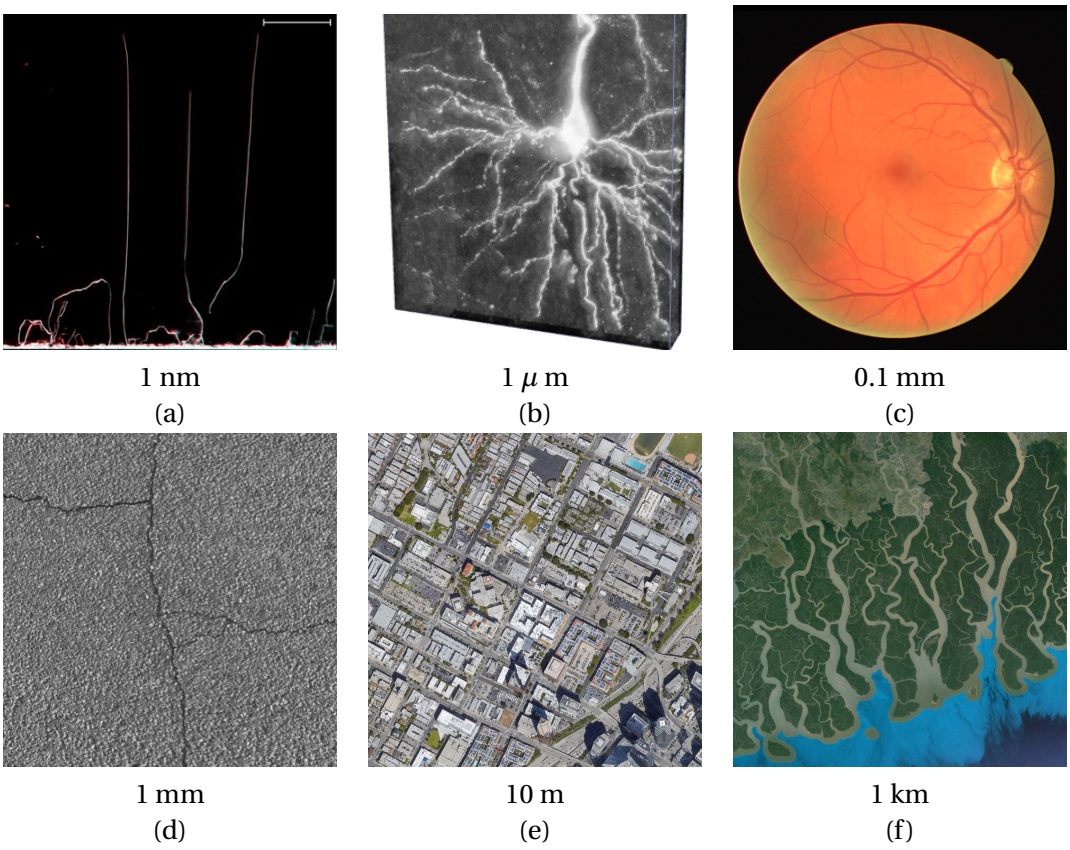
1.1 Motivation

The recent advances in imaging techniques let one obtain vast amounts of data that play a crucial role in different applications. However, in order to analyze them and draw conclusions, one must first detect the structures of interest, which can be difficult given the typically large amounts of data, which has to be examined. For this reason, recent years have seen an increase in interest in automated methods, which can provide reconstructions that normally would take days or weeks to complete manually. Below, we present several domains, where such tools prove to be especially useful.

1.1.1 Biomedical Sciences

There are multiple examples of line-like structures in living organisms, perhaps the most prominent ones being neurons and blood vessels.

Neuronal structures can be captured with electron microscopy (EM) as well as light microscopy images, an example of which is shown in Fig 1.1(b). Their size varies from 4 to 100 microns in diameter and their length is in the range of several to tens centimeters. It is estimated that the human brain is composed of more than 100 billion neurons, which can be classified to different



1 nm (a) | 1 μ m (b) | 0.1 mm (c)

1 mm (d) | 10 m (e) | 1 km (f)

Figure 1.1 – **Curvilinear structures** are widely present in nature and man-made systems. Above we present several examples of such structures along with scale on which they appear: (a) dislocations in crystals in Transmission Electron Microscopy images [82] (b) neurons in mouse brain in brighfield microscopy 3D scans (c) retinal blood vessels in fundus images [103] (d) cracks in materials [127] as well as (e) roads [8] and (f) rivers in satellite images (courtesy of Google Maps)

categories depending on their morphology. This information is usually recovered through laborious manual reconstruction, which is recognized as one of the major bottlenecks in neuroscience research. To give a rough estimate, annotating one cell can take up to a week and can be usually done only by an expert. Neuron's morphology is also crucial for describing its electrical properties, which in turn are critical for simulating networks of interacting neurons. The importance of this topic is emphasized by the recent initiatives, whose aim is to develop automated methods for neuron reconstruction. These include, amongst others, the DIADEM challenge [7] and the BigNeuron project [97]. Additionally, one part of the biggest European research project, The Human Brain Project [63], is focused on this task. Collecting a large number of reconstructions may reveal new types of neurons, allow for better visualizations and more accurate simulations of the complex processes taking place in the brain. Eventually, it will be crucial for developing new computing technologies and even more importantly for

understanding and treating brain diseases.

Another application of delineation in the biomedical sciences is detection of blood vessels. Examples include blood vessels in the brain or in the liver, that can be imaged using magnetic resonance imaging (MRI) or computed tomography (CT) scans, as well as retinal blood vessels in fundus images (1.1(c)). Similarly to neurons, vessels diameter can vary significantly from only 8 μm to about 25 mm. Reconstruction of vessel networks is a necessary step in simulations of blood flow and in diagnostics of vascular diseases. Certain pathologies (*e.g.* calcifications, stenoses and diabetes) alter the morphology of blood vessels and those changes can be seen in angiograms (certain type of X-ray images) of coronary arteries [64] or fundus images depicting retina [44, 37]. Automatic reconstruction could enable faster diagnosis and provide quantitative measures of severeness. Moreover, precise determination and visualization of the vascular anatomy or pathology may be helpful during surgery planning and navigation [54, 13].

1.1.2 Remote Sensing

One of the most common man-made curvilinear structures are roads (Fig. 1.1(e)). In order to plan the driving route efficiently in their dense network, one needs accurate maps. This is even more true when it comes to autonomous vehicles, whose introduction seems imminent.

The rate at which infrastructure changes in the fast-growing cities around the world means that it is getting increasingly difficult to update maps frequently enough and to verify changes. For example, it is estimated that annotating roads within 1 km^2 region takes 8 hours [65]. At the same time it is now possible to obtain new satellite images of very high resolution almost on a daily basis; Sentinel-1 [2] — satellites developed by the European Space Agency — capture images of the entire Earth every 6 days. In order to make a full use of this data it is necessary to develop automated detection methods, which would allow for much more frequent map updates.

Increasing number of humanitarian crises sparked interest in using computer-assisted methods for natural disaster mapping and emergency management [3]. Unfortunately, the regions that are the most prone to such disasters are also often those that have not been mapped well. Obtaining maps of such areas could considerably decrease the response time in case of emergency. When a disaster occurs, it is also extremely important to identify which parts of the infrastructure were affected the most in order to coordinate the rescue operations [89]. Organizations such as Humanity& Inclusion [1] organize so called *mapathons*, where volunteers create maps for regions that were not well covered. However, this is not a scalable and sustainable solution and leveraging automated methods for this task could bring great benefits to the community.

1.1.3 Materials Science

Cracks, such as the one shown in Fig. 1.1(d) are flaws in materials that result from mechanical stress. They can emerge during manufacturing or exploitation. Analysis of evolving crack patterns is the basis for fracture mechanics. Its aim is to identify which cracks are stable (and hence "safe") and which of them propagate and may pose danger of material failure. Examples of possible applications include inspection of buildings, bridges or jet engine [83, 19]. In order to quantify the crack evolution process, images of the monitored part are taken over time and crack size and shape is evaluated and tracked.

Cracks are not the only linear structures appearing widely in materials. Dislocations are defects in crystalline lattice, which, being thermodynamically unavoidable, determine a large range of mechanical properties of materials including their strength [81, 82]. The modern electron microscopes allow one to see many materials on atomic scale (Fig. 1.1(a)), which in turn allows the individual dislocations to be studied.

While there exists software for cracks and dislocations simulation, so far there have been only few attempts at computer-assisted reconstruction from real images, which could improve the physical models. Automated approaches would speed up the analysis and enable more precise quantification of the properties of interest.

1.2 Challenges

There are numerous challenges associated with the detection of linear structures, which we present in this section and in the next section we discuss how our work addresses them.

1.2.1 Occlusions and Noise

One of the most challenging conditions for delineation are in the presence of occlusions and noise. To give an example, big parts of roads in the satellite images are often occluded by trees as seen in Fig. 1.2(a). Furthermore, structures depicted in biomedical images may suffer from uneven staining of the samples and noise in the background originating from the acquisition process (Fig. 1.2(b)). Human observers are very good at predicting *e.g.* the path of a road despite these problems, due to very strong prior on road structure and shape. Thanks to this, even if part of the street disappears under the trees and continues afterwards, one infers that most probably the road is occluded, as it is usually continuous. The same applies to neuronal structures — even if part is missing in the image due to imaging or sample preparation artifacts, an experienced neuroscientist can deduce which branches are connected. In order to mimic this process in automated systems, one needs to take into account a large context

and introduce prior knowledge in the algorithms.

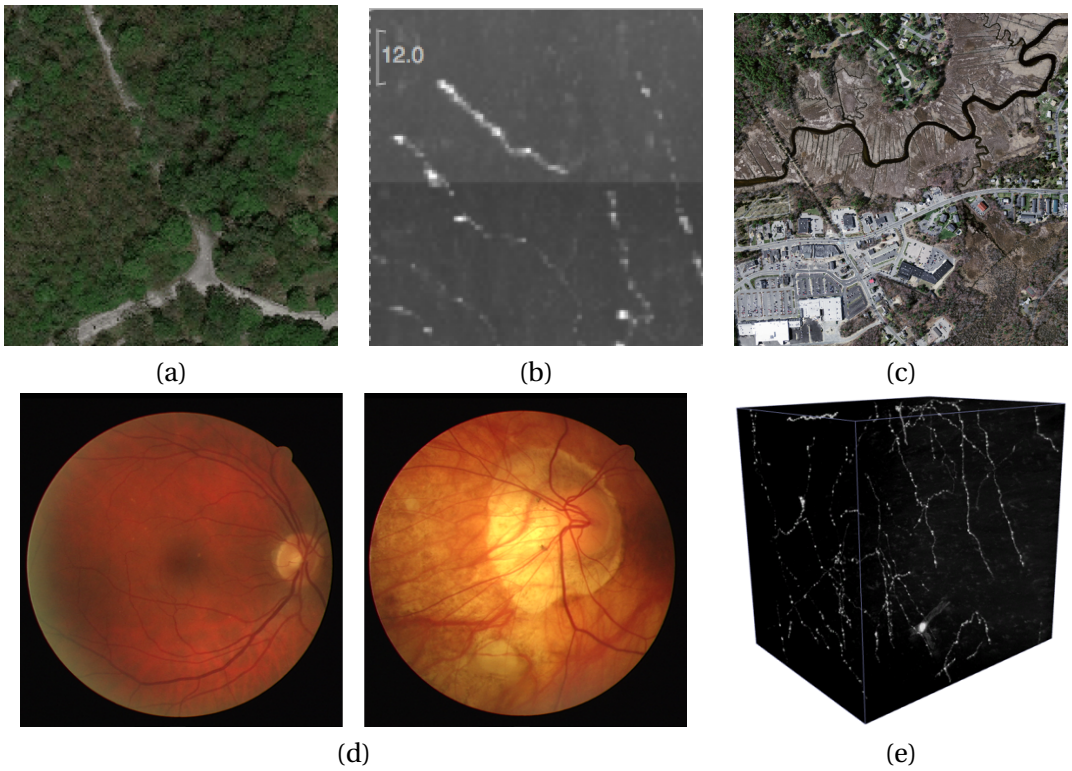


Figure 1.2 – **Challenges.** There are multiple challenges associated with delineation. (a) Occlusions: trees occluding roads in satellite images may cause discontinuities in predictions. (b) Imaging artifacts: noise in the background and uneven staining of the neuron sample lead to gaps in the image intensity (c) Variety of appearances: rivers and roads are both curvilinear structures but we may be interested in only one class of objects. (d) Variety of appearances: in biomedical imaging the appearance may vary significantly depending on the subject (healthy: left vs. diabetic: right) as well as imaging equipment. (e) Data size: most of microscopic images are intrinsically 3D and are often impossible to be viewed and processed at once. This cube represents one of around 1/20 000th of the whole mouse brain.

1.2.2 Variability of Appearances

Another important issue is the large intraclass variability of appearances of such structures. To give an example, roads in satellite images include both multi-lane highways and narrow backcountry paths. To make things even more confusing, large interclass similarity increases the risk of false detections. There are examples of linear structures such as parking lots, small driveways or even rivers (Fig. 1.2(c)), which may resemble roads but should not be detected. In the biomedical domain the variability may arise from the multitude of equipment models used for image acquisition, changes to the structure appearance due to some diseases (the difference in blood vessels in a healthy subject vs. a patient with retinopathy is shown in

Fig. 1.2(d)) or, in the case of neurons - imaging different parts of the brain, which contain different classes of neurons. In order to build Machine Learning systems that generalize well to a wide range of appearances of such structures, but at the same time can distinguish between false positives, one needs to supply a large amount of examples during training procedure. This brings us to the next challenge.

1.2.3 Lack of Annotated Data

Machine Learning (and even more so Deep Learning) methods require considerable amounts of annotated ground-truth data for training. While in some domains (e.g. natural image classification) this is not hard to obtain (already there exist multiple datasets comprising of millions of annotated examples), in the task of segmentation it is much more difficult and tedious to generate precise labelling. It becomes even more apparent when dealing with structures in 3D. To give an example, a recent study [51] has shown that it takes around 1 hour to annotate a small stack of axons. In order to decrease the workload associated with annotation, several alternatives have been proposed. In the task of road detection, the readily available maps have been registered with satellite images and used to train automated algorithms. The disadvantage of this approach is that the maps may be noisy. In the biomedical domain the problem is more complex because there does not exist a way to obtain the labels "for free". Hence, so far, most of effort in this direction was focused on improving the visualization tools and making the annotation process more convenient. One alternative that is explored in this thesis is using Active Learning (AL), which aims at reducing annotation effort by selecting only the most informative image regions for labelling.

1.2.4 Data Size

Recent advances in microscopy allow one to obtain large amounts of high-quality data. This resulted in a significant increase in size of stored datasets. To give an example, the whole mouse brain dataset, which Blue Brain Project shared with us, is 12 000×24 000×36 000 voxels in size and it occupies 300 GB in compressed format. Similarly, satellite images acquired today on a global scale have much higher resolution than they used to and they are captured much more frequently. Consequently, since its launch in 2014 Sentinel satellites have acquired about 25 petabytes of images [126]. One of the challenges is visualizing this vast amount of data, which usually cannot be viewed all at once. Similarly, the data are often processed in small chunks, as floating-point representation would require tens of gigabytes of GPU memory. Microscopic images are usually 3D stacks and although it is possible to analyze them slice by slice, it is at expense of information loss. Any future reconstruction methods oriented towards biomedical applications should be therefore capable of analyzing large 3D stacks.

1.3 Contributions

The main aim of this thesis is to develop methods for robust and efficient delineation of curvilinear structures. To this end we investigated two research directions: building a detection pipeline and reducing the annotation effort involved in preparing the ground-truth data. The main insight of our research was that by incorporating the notion of topology or geometry, we were able to considerably improve the results at smaller annotation cost. The proposed methods improve every step of the delineation process, from obtaining ground-truth data efficiently, to topologically correct segmentation and graph representation, to decreasing the time spent on proofreading the result. As a result, they enable reconstruction of wide variety of curvilinear structures within minutes and they are a step forward in analysis of this vast amounts of data.

1.3.1 Topology-aware Segmentation Loss

The first step of most delineation algorithms is segmentation of the structure of interest. The recent approaches to this task have focused on crafting better features that are later used for pixel classification or, in case of Deep Learning, finding better network architectures. However, the conventional solutions still rely on the habitual pixel-wise loss used during training, which is unable to capture the structural properties of curvilinear networks.

Contrary to that, we propose a novel loss term that computes the training loss not only at the pixel level, but also by looking at higher-level features. To this end, we exploit a VGG network [100] pre-trained on ImageNet [21] to compute features that capture topology of the linear structures and compare prediction and ground-truth using this additional statistics. As a result, the topology loss differentiates structural differences better than binary cross-entropy. We test the proposed approach for segmentation of roads, cracks and neuronal membranes and we demonstrate that it considerably improves the resulting delineations, not only in terms of pixel accuracy, but also topology.

1.3.2 Multi-task Learning for Graph Extraction

In many applications the pixel-wise segmentation of the network of interest is not enough and its underlying graph representation is essential to infer connectivity information. In order to extract it from the probability map, most of the approaches first perform image segmentation and then sample from it multiple possible paths, which are then classified using a separate classifier. This is not an easy task taking into account the large amount of possible scenarios of positive and negative paths.

We therefore exploit the fact that the path classification shares similarities with segmentation task and we jointly train the network that segments linear structures and classifies the proposed connected paths. We validate this approach on a dataset od roads spanning 15 cities around the world and on a 3D mouse axons dataset.

1.3.3 Active Learning for Delineation

Reconstruction of curvilinear structures has been shown to greatly benefit from supervised Machine Learning. However, it usually requires laborious and time-consuming manual annotation of ground-truth data by an experienced annotator. We present two methods based on Active Learning (AL) that have a potential to considerably accelerate this process. While most of the AL methods are developed for general purposes and as such they do not take into account specificity of delineation, both of our approaches exploit geometry-consistent queries, which allows us to account for geometrical constraints. They operate on graph-representation of the curvilinear network and identify parts of the graph that should be annotated first.

In our first approach we ensure *local* consistency of the classified graph edges and we only select the samples that violate this constraint for annotation. Additionally, we introduce feature-space density measures, which allows for exploiting the unlabeled data distribution to further reduce the annotation effort.

Our second approach addresses both ground-truth annotation of training data and proofreading the resulting reconstruction by taking into account the influence of a potential misclassified edges on the resulting *global* delineation graph. In an Active Learning context, we identify parts of linear structures that should be annotated first in order to train a classifier effectively. In proofreading settings, we similarly find regions of the resulting reconstruction that should be verified first to obtain a nearly-perfect result. To this end, we quantify how the reconstruction graph changes when one of its edges changes its weight.

We apply our two approaches to detection and reconstruction of curvilinear networks in aerial 2D and microscopic 3D images. We show that our methods of identifying samples in ambiguous image regions allow one to reduce the training set size by up to 80% without compromising the quality of reconstruction.

1.4 Outline

The reminder of this thesis is organized as follows: we begin by discussing related work on detection of curvilinear structures, including both segmentation and graph-based methods. We also present previous work on general Active Learning. In Chapter 3, we present a novel loss

term that is particularly suitable for segmentation of curvilinear structures and which carries a notion of topology in the delineation. In the following chapter we show how multi-task learning can be exploited to simultaneously perform segmentation and obtain the connectivity of a curvilinear network. In Chapters 5 and 6 we present two Active Learning methods that exploit the characteristics of curvilinear networks to select the most informative samples for annotation. We conclude this thesis with a summary of our findings and ideas for future work.

2 Literature Overview

The literature related to detection of curvilinear networks is very diverse, as many methods are application-specific and investigate only one type of structure — roads, neurons, blood vessels etc. It includes both methods which aim at segmentation or centerline detection, as well as those that produce a full graph representation, which is especially important in neuroscience or cartography. We first discuss the relevant literature concerning pixel-wise classification methods that were used in the context of curvilinear structures. Then, we give an overview of approaches based on extracting their graphical representation. Finally, we discuss general Active Learning methods that aim at reducing the annotation effort and the more specific ones that were used in Computer Vision domain.

2.1 Segmentation of Curvilinear Structures

Delineation algorithms can rely either on hand-crafted or on learned features. Early approaches relied on the former and the most prominent group are Hessian-based methods [104]. They postulate "a priori" model, which assumes that curvilinear structures have cross-sectional Gaussian intensity profile and a local principal direction. The principal direction is then used to guide the delineation *e.g.* by using snakes [76]. Another family of unsupervised approaches uses hand-designed filters that find specific patterns. Examples include Optimally Oriented Flux (OOF) [55] and Multi-Dimensional Oriented Flux (MDOF) [108], its extension to irregular structures. Jacob *et al.* [41, 5] propose 2D and 3D steerable filters that maximize Canny-like criteria to respond to a particular type of features. The great strength of unsupervised methods is that they do not require training data but at the cost of struggling with very irregular structures at different scales along with the great variability of appearances and artifacts.

In such challenging situations, learning-based methods have an edge and several approaches

have been proposed over the years. For example, the approach of [124] combines Haar wavelets with boosted trees, while that of [39] performs SVM binary classification on spectral features. Becker *et al.* [9] learn jointly features and a classifier for segmentation of biomedical structures relying on a Gradient Boosting framework. In [102], the classifier is replaced by a regressor that predicts the distance to the closest centerline, which enables estimating the width of the structures. It is further improved in [101] by projecting part of the prediction to its nearest neighbour in training set in order to eliminate mistakes such as discontinuities and spurious detections. Gu and Cheng [36] propose an approach that can be combined with any initial segmentation in order to boost thin filamentary structures, which otherwise can be easily missed.

In more recent work, Decision Trees and SVMs have been replaced by Deep Networks. For the purpose of road delineation, this was first done in [70], directly using image patches as input to a fully connected neural net. While the patch provided some context around the linear structures, it was still relatively small due to memory limitations. With the advent of Convolutional Neural Networks (CNNs), it became possible to use larger receptive fields. In [30], CNNs were used to extract features that could then be matched against *words* in a learned dictionary. The final prediction was made based on the votes from nearest neighbors in the feature space. A fully-connected network of [70] was replaced by a CNN in [69] for road detection. In [67] a differentiable Intersection-over-Union loss replaced a conventional binary-cross entropy to obtain a road segmentation, which is then used to extract the graph of the road network. The Holistically-Nested Edge Detector [118] (HED) is a method for multi-scale deep end-to-end learning, where the edges are hierarchically integrated at different scales contrary to many previous methods that used hand-crafted features and did not refine edges appearing on different scales efficiently. Its extension to 3D data [68] was successfully introduced in the task of vasculature detection.

In the biomedical field, the VGG network [100], pre-trained on real images, has been fine-tuned and augmented by specialized layers to extract blood vessels [62]. Similarly, the U-Net [88], has been shown to give excellent results for biomedical image segmentation and is currently among the methods that yield the best results for neuron boundaries detection in the ISBI'12 challenge [6]. It is composed of an encoder, that is, a contracting path that captures context, followed by an expanding decoder, which localizes objects. It comprises a series of convolutional layers with either increasing or decreasing number of channels, interleaved with pooling operations for the encoding layers and up-convolutions for the decoding layers. Skip connections between corresponding layers in contracting and expanding paths provide high-resolution features to the latter.

Different variants of the U-Net were proposed to extract roads segments or their centerlines. In [121] a conventional U-Net architecture was augmented with residual connections, shown

to facilitate backpropagation to deeper layers and applied to road extraction. In order to extract centerlines from segmentation, a two-step cascaded U-Net was proposed in [15], where the first stage outputs segmentation and the next one refines centerlines. Manual labelling of satellite images is very time-consuming and the maps available online and created by volunteers are often noisy. In [46] it was shown that training the U-Net on a large dataset consisting of noisy annotations obtained from the online maps reaches the same performance as when using smaller but more accurate ground-truth. Although U-Net provides structured output, in all aforementioned cases the training loss is computed by inspecting per-pixel discrepancies between the prediction and ground-truth. As such, it does not capture small mistakes that may significantly impact the overall structure of the result. In Chapter 3, we present a topology loss that accounts also for higher-level statistics of ground-truth.

As mentioned in the previous chapter, linear structures present certain geometric characteristics that can be incorporated to train the detector. To give an example, one way of imposing topological knowledge in the output of curvilinear structure detectors is by introducing higher-order Conditional Random Fields (CRF). In [115], priors are computed on cliques of connected superpixels likely to be part of road-like structures. Unfortunately, due to the huge number of potential cliques, it requires sampling and using hand-designed features to obtain superpixels likelihood of belonging to the positive class. Similarly, in [29] CRF was combined with the output of Neural Network in the task of retinal blood-vessel detection to account not only for discriminative pixel probabilities, but also long-range pixel interactions. Orlando and Blaschko [80] learn the parameters of CRF using structured output SVM to learn richer potentials that improve the detection of thin blood vessels. Another way of accounting for long-range relationships between parts of the curvilinear networks is to treat it as a graph, which is described in the next subsection.

2.2 Graph-based Reconstruction of Curvilinear Structures

Graph-based reconstruction algorithms have recently shown superior performance compared to methods based solely on segmentation. They recover not only the geometry of the problem, but also the correct connectivity, which is crucial in applications such as neuroscience.

The robustness of the graph-based methods was further improved by applying supervised Machine Learning techniques. Many of them consist of three steps:

1. A local (pixelwise) measure of tubularity is computed using hand-designed filters or assigned by a classifier as described in the previous section.

2. Then candidate segments (paths) are sampled from the segmentation and they are assigned a likelihood of being a curvilinear structure. The paths constitute edges of an

over-complete graph.

3. Finally, the low-scoring path proposals are filtered out from the over-complete graph using a set of heuristics or optimization, resulting in the final reconstruction.

For example, [72] first performs local pixel classification, detects elongated road candidates using a shortest-path strategy and finally selects paths that minimize the energy in a CRF. Similarly, Wegner *et al.* [116] formulate the problem as finding the set of shortest paths between cliques of superpixels and solving MAP inference where the path cost is assigned by a classifier. Breitenreicher *et al.* [11] use two separate classifiers to first assign pixel probabilities, then they generate centerline candidates and finally verify them using a tree-based model trained with hand-designed features such as the distribution of intensities and the tubularity along candidate paths.

For certain structures, knowing global prior information about their topology allows us to impose certain constraints to obtain a more realistic graph representation. To give an example, neurons are always tree-like structures (they do not form loops) and as a result many approaches to neuron reconstruction are based on finding Minimum Spanning Tree (MST) [84, 119]. They usually first produce an over-complete representation of the network connecting set of previously found vertices (high-probability pixels) and prune edges producing minimum-cost tree. The edge cost depends on its appearance and may be computed by integrating the values of probability/tubularity along it [111] or assigned by a classifier [11]. As MST spans all possible vertices and in reality not all of them may be in the solution, a pre-defined heuristic can be used to further prune some edges and vertices [111]. Turetken *et al.* first replaced pruning heuristics with Integer Program formulation [110] and then extended it to loopy structures [109]. As it is an NP-complete problem, it uses a branch-and-bound algorithm for finding the optimal solution and is computationally expensive. In Chapter 6 we present a faster formulation of the same method, which considerably decreases the size of an optimization problem and allows for interactive applications. Glowacki *et al.* [33, 34] extended the approach of [109] to take into account also temporal information and ensure that the resulting graphs are consistent over time. Their approach also allows for automatic detection of changes in consecutive graphs.

Another approach to model higher-level statistics of curvilinear structures is to represent them as a sequence of connected short linear segments, which can be accomplished using a Marked Point Process [105, 106, 12, 45]. They aim at minimizing an energy function that uses various components, such as homogeneity of the pixel values, graph connectivity, edge orientation or line width. However, the inference involves Reversible Jump Markov Chain Monte Carlo, which is computationally expensive and relies on a very complex objective function. Mattyus *et al.* [65] model road segments as centerlines with corresponding width and improve online

road maps by formulating the problem as a Markov Random Field. There, an energy function captures the appearance of roads, edge information, detected cars, smoothness of the segments and continuous width.

Up until recently there were no Deep Learning approaches to graph-based reconstruction of curvilinear structures, as training the neural network that can output a graph is a non-trivial task. Mattyus *et al.* [67] proposed an approach that as a post-processing step enhances the graph by reasoning about missing connections overlaid on the image and applying a set of heuristics to remove false detections. However, it also requires training a separate model for connection classification and setting many hyperparameters. Another two-step method [112] is inspired by pose-estimation approaches and for every image patch it produces a heatmap of border points that are connected to the center point. Zhou *et al.* [125] also infer the presence or lack of connection between two vertices using a Siamese network. Both approaches, using this connectivity map and the previously obtained semantic segmentation, they connects the connected points using Dijkstra's algorithm [23]. All the described methods train two separate classifiers to first obtain a segmentation map and then predict connections between two points. In constrast, in Chapter 4 we bridge the gap between those two tasks and we introduce a multi-task learning approach to simultaneous segmentation *and* path classification.

The only attempt to directly output a complex graph from a Deep Net was presented in [8], where a CNN was trained to output the direction in which tracing should proceed. A separate network was trained to predict end-points. A vessel extraction method based on Reinforcement Learning was proposed in [122]. There the artificial agent is trained to interact with the surrounding environment and allows for tracing arbitrarily long vessels, but without bifurcations.

2.3 Active Learning

Active Learning (AL) is predicated on the idea that the learning algorithm can actively choose the next training instance from a large unlabelled pool of samples that would be the most profitable to label next. Iterating this process can drastically reduce the need for further human annotation since only the most informative ones are considered. This has been demonstrated in applications ranging from Natural Language Processing to Bioinformatics in which unlabeled data is readily available but annotation is expensive [93].

All such AL methods require a criterion for sample selection. The most popular one is sample *uncertainty*, usually defined as the proximity to the classifier's decision boundary. When the classifier returns a probability, this can be evaluated in terms of entropy [95]. In practice, Uncertainty Sampling can be incorporated into most supervised learning methods such as

SVMs [107], Boosting [38] and Neural Networks [17].

Another family of AL algorithms called *query-by-committee* [18] uses different automated "experts" to assign potentially different labels to each sample. Those for which the disagreement is greatest are prime candidates for manual annotation.

Most practical AL algorithms allow the human to annotate batches of samples before retraining the classifier. This spares them having to wait for a potentially lengthy computation to finish between each one of their interventions. However, Uncertainty Sampling, as described above, in batch mode can easily end up querying outliers and redundant instances, which is inefficient. This is usually addressed by considering not the only the information gain potentially delivered by labelling each individual sample, but also the representativity of each batch, which is accomplished by *density-based* methods. In [95] Settles and Carven introduce a information density-weighted framework, which favours samples that are not only uncertain but also representative for the underlying distribution. The main problem associated with this approach is finding the weighting of the two terms. Li and Guo [56] propose choosing weights at each iteration that would minimise the future generalisation error. This approach is however computationally expensive, as it requires recomputing the underlying model many times and additionally may lead to overfitting.

Most of the methods discussed above originate from fields other than Computer Vision. They rarely exploit the contextual or spatial relations that are prevalent in images except for a few exceptions. In [99] contextual image properties are used to find the image regions that would convey the most information about other uncertain areas with which they are contextually related. In video segmentation [27], the obtained labels are propagated in a semi-supervised manner on a graph consisting of spatial, temporal and prior edges. Then, the most uncertain frame is selected for the next annotation. As we show in Chapter 5, propagating information *after* preliminary classification and computing the uncertainty only after that step has got an advantage over estimating informativness based on the result of only one classifier. Jain and Grauman [42] propose a foreground-background propagation model, which favours labelling uncertain images that are also influential (foreground segmentation masks can be easily transformed to unlabelled images) and diverse. The AL approach to segmenting CT scans of [40] incorporates context in terms of generative anatomy models. In [113] AL for semantic segmentation is modeled as a pairwise CRF capturing the properties of individual superpixels as well as spatial similarities between neighbouring instances. The next query is selected based on its potential to induce label change in other variables. The notion of geometric uncertainty for segmentation is introduced in [50] to indicate ambiguous superpixels, which are usually localized on the object's border. Like our algorithm of Chapter 5, it relies on exchanging probability values between neighbouring superpixels, but it does not account for dataset diversity and, unlike ours, it requires setting the number of probability propagation iterations

beforehand to ensure good performance.

Some of work related to Active Learning [114, 94, 61] investigated the actual cost associated with obtaining each label, contrary to methods described before, which assumed that this cost is constant. There, the problem is formulated as minimizing the total cost of annotation, which can be estimated using trained cost prediction model.

3 Beyond the Pixel-wise Loss for Topology-Aware Delineation

Segmentation of curvilinear structures is the first step of many delineation algorithms and constitutes the basis for many subsequent post-processing steps. With the advent of Deep Learning, many current approaches on automatic delineation have focused on finding more powerful deep architectures, but have continued using the habitual pixel-wise losses such as binary cross-entropy. In this chapter we show that pixel-wise losses alone are unsuitable for this problem because of their inability to reflect the topological impact of mistakes on the final prediction. We propose a new loss term that is aware of the higher-order topological features of linear structures. We also introduce a refinement pipeline that iteratively applies the same model over the previous delineation to refine the predictions at each step, while keeping the number of parameters and the complexity of the model constant.

When combined with the standard pixel-wise loss, both our new loss term and our iterative refinement boost the quality of the predicted delineations, in some cases almost doubling the accuracy as compared to the same classifier trained with the binary cross-entropy alone. We show that our approach outperforms state-of-the-art methods on a wide range of data, from microscopy to aerial images.

Part of the material presented in this Chapter appeared in [73].

3.1 Introduction

Automated delineation of curvilinear structures, such as those in Fig. 4.1(a, b), has been investigated since the inception of the field of Computer Vision in the 1960s and 1970s. Nevertheless, despite decades of sustained effort, full automation remains elusive when the image data is noisy and the structures are complex. As in many other fields, the advent of Machine Learning techniques in general, and Deep Learning in particular, has produced substantial advances, in large part because learning features from the data makes them more robust to appearance

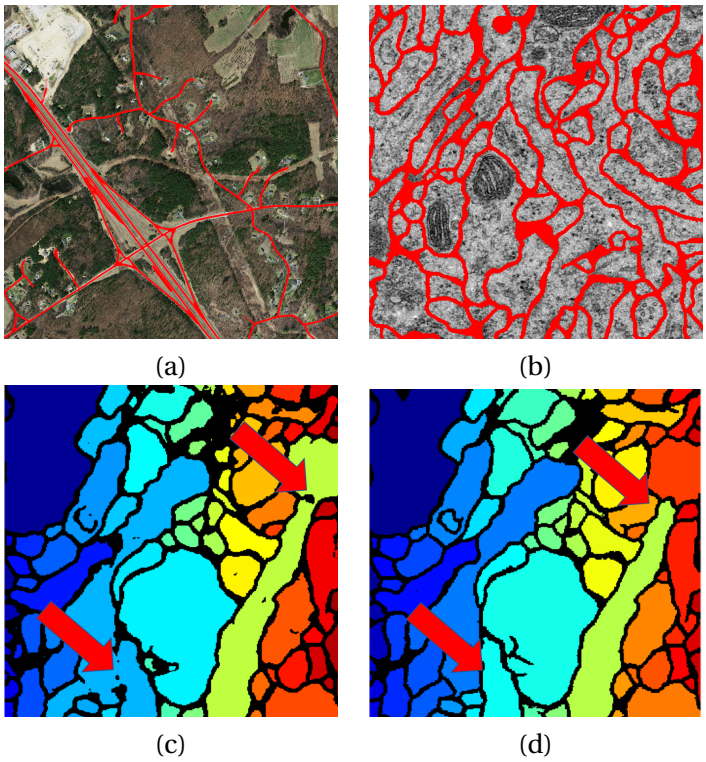


Figure 3.1 – **Segmentation of curvilinear structures.** (a) Detected roads in an aerial image. (b) Detected cell membranes in an electron microscopy (EM) image. (c) Segmentation obtained after detecting neuronal membranes using [88] (d) Segmentation obtained after detecting membranes using our method. Our approach closes small gaps, which prevents much bigger topology mistakes.

variations [30, 71, 102, 115].

However, all new methods focus on finding either better features to feed a classifier or more powerful deep architectures, while still using a pixel-wise loss such as binary cross-entropy for training purposes. Such loss is entirely local and does not account for the very specific and sometimes complex topology of curvilinear structures penalizing all mistakes equally regardless of their influence on geometry. As a shown in Fig. 4.1(c,d) this is a major problem because small localized pixel-wise mistakes can result in large topological changes.

In this chapter, we show that supplementing the usual pixel-wise loss by a *topology loss*, which promotes results with appropriate topological characteristics, yields a substantial performance increase without having to change the network architecture. In practice, we exploit the feature maps computed by a pretrained VGG19 [100] to obtain high-level descriptions that are sensitive to linear structures. We use them to compare the topological properties of the ground truth and the network predictions and estimate our topology loss.

In addition to this we exploit iterative refinement freamework, which is inspired by the recurrent convolutional architecture of Pinheiro and Collobert [86]. We show that, unlike in the recent methods [78, 98], sharing the same architecture and parameters across all refinement steps, instead of instantiating a new network each time, results in state of the art performance and enables keeping the number of parameters constant irrespectively of the number of iterations. This is important when only a relatively small amount of training data is available, as is often the case in biomedical and other specialized applications.

We evaluate our approach on a range of datasets comprising of both natural images (roads and cracks) as well as biomedical (neuronal membranes) using not only conventional segmentation metrics, but also topology-oriented measures.

3.2 Method

We use the fully convolutional U-Net [88] as our trainable segmentation model, as it is currently among the best and most widely used architectures for delineation and segmentation in both natural and biomedical images. The U-Net is usually trained to predict the probability of each pixel of being a linear structure using a standard pixel-wise loss. As we have already pointed out, this loss relies on local measures and does not account for the overall geometry of curvilinear structures, which is what we want to remedy.

In the remainder of this section, we first describe our topology-aware loss function, and we then introduce iterative procedure to recursively refine the predictions.

3.2.1 Notation

In the following discussion, let $\mathbf{x} \in \mathbb{R}^{H \cdot W}$ be the $W \times H$ input image, and let $\mathbf{y} \in \{0,1\}^{H \cdot W}$ be the corresponding ground-truth labeling, with 1 indicating pixels in the curvilinear structure and 0 indicating background pixels.

Let f be the U-Net parameterized by weights $\mathbf{w}$, shown in Fig. 3.2. The output of the network is an image $\hat{\mathbf{y}} = f(\mathbf{x}, \mathbf{w}) \in [0,1]^{H \cdot W}$.[1] Every element of $\hat{\mathbf{y}}$ is interpreted as the probability of pixel i having label 1: $\hat{\mathbf{y}}_i \equiv p(Y_i = 1 \mid \mathbf{x}, \mathbf{w})$, where Y_i is a random Bernoulli variable $Y_i \sim \mathrm{Ber}(\hat{\mathbf{y}}_i)$.

[1] For simplicity and without loss of generality, we assume that $\mathbf{x}$ and $\hat{\mathbf{y}}$ have the same size. This is not the case in practice, and usually $\hat{\mathbf{y}}$ corresponds to the predictions of a cropped area of $\mathbf{x}$ (see [88] for details).

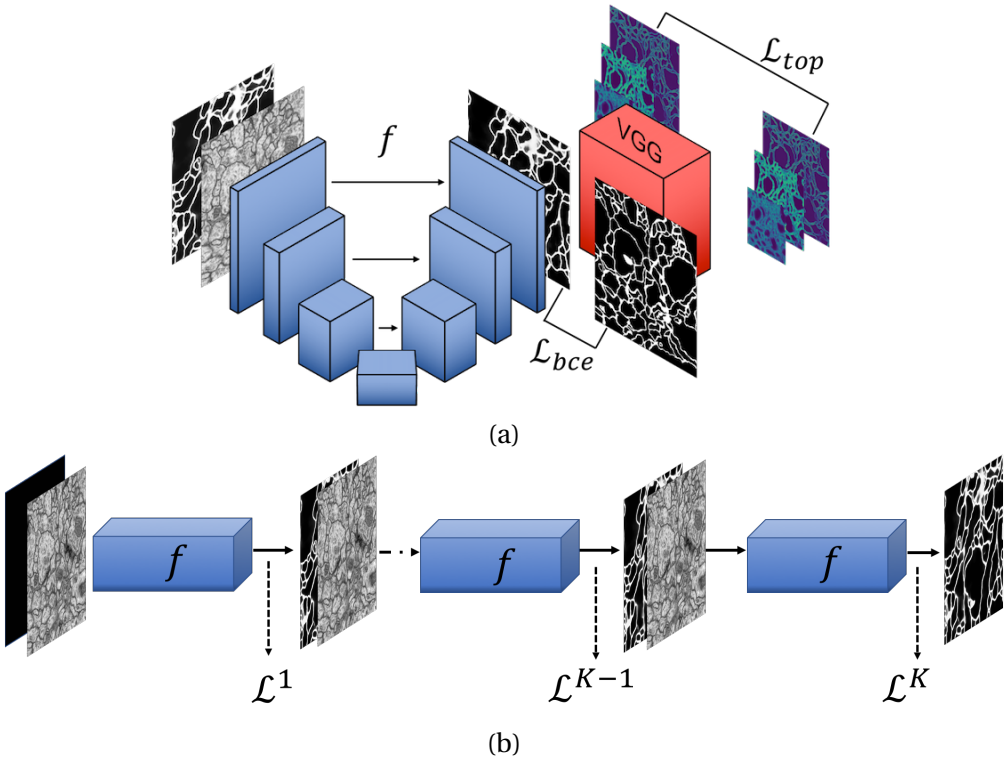


Figure 3.2 – **Network architecture.** (a) We use the U-Net for delineation purposes. During training, both its output and the ground-truth image serve as input to a pretrained VGG network. The loss $\mathscr{L}_{top}$ is computed from the VGG responses. The loss $\mathscr{L}_{bce}$ is computed pixel-wise between the prediction and the ground-truth. (b) Our model iteratively applies the same U-Net f to produce progressive refinements of the predicted delineation. The final loss is a weighted sum of partial losses $\mathscr{L}^k$ computed at the end of each step.

3.2.2 Topology-aware loss

In ordinary image segmentation problems, the loss function used to train the network is usually the standard pixel-wise binary cross-entropy (BCE):

$$\mathscr{L}_{bce}(\mathbf{x},\mathbf{y},\mathbf{w}) = -\sum_i \left[(1-\mathbf{y}_i)\cdot\log(1-f_i(\mathbf{x},\mathbf{w})) + \mathbf{y}_i\cdot\log f_i(\mathbf{x},\mathbf{w})\right].$$

Even though the U-Net computes a structured output and considers large neighborhoods, this loss function treats every pixel independently. It does not capture the characteristics of the topology, such as the number of connected components or number of holes. This is especially important in the delineation of thin structures: as we have seen in Fig. 4.1(c, d), the misclassification of a few pixels might have a low cost in terms of the pixel-wise BCE loss, but have a large impact on the topology of the predicted results.

Therefore, we aim to introduce a penalty term in our loss function to account for this higher-

order information. Instead of relying on a hand-designed metric, which is difficult to model and hard to generalize, we leverage the knowledge that a pretrained network contains about the structures of real-world images. In particular, we use the feature maps at several layers of a VGG19 network [100] pretrained on the ImageNet dataset as a description of the higher-level features of the delineations. We chose VGG for this purpose because its lower layers have been shown to respond strongly to line-like patterns and have been successfully used to transfer the appearance and arrangements of objects from one image to another [31]. Our new penalty term tries to minimize the differences between the VGG19 descriptors of the ground-truth images and the corresponding predicted delineations:

$$\mathscr{L}_{top}(\mathbf{x},\mathbf{y},\mathbf{w}) = \sum_{n=1}^{N} \frac{1}{M_n W_n H_n} \sum_{m=1}^{M_n} \left\| l_n^m(\mathbf{y}) - l_n^m(f(\mathbf{x},\mathbf{w})) \right\|_2^2 , \tag{3.1}$$

where l_n^m denotes the m-th feature map in the n-th layer of the pretrained VGG19 network, N is the number of convolutional layers considered and M_n is the number of channels in the n-th layer, each of size $W_n \times H_n$. $\mathscr{L}_{top}$ can be understood as a measurement of the difference between the higher-level visual features of the linear structures in the ground-truth and those in predicted image. These higher-level features include concepts such as connectivity or holes that are ignored by the simpler pixel-wise BCE loss. Fig. 3.3 shows examples where the pixel-wise loss is too weak to penalize properly a variety of errors that occur in the predicted delineations, while the loss $\mathscr{L}_{top}$ correctly measures the topological importance of the errors in all cases: it penalizes more the mistakes that considerably change the structure of the image and those that do not resemble linear structures.

The reason behind the good performance of the VGG19 in this task can be seen in Fig. 3.4. Certain channels of the VGG19 layers are activated by the type of elongated structures we are interested in, while others respond strongly to small connected components. Thus, minimizing $\mathscr{L}_{top}$ strongly penalizes generating small false positives, which do not exist in the ground-truth, and promotes the generation of elongated structures. On the other hand, the shape of the predictions is ignored by $\mathscr{L}_{bce}$.

In the end, we minimize

$$\mathscr{L}(\mathbf{x},\mathbf{y},\mathbf{w}) = \mathscr{L}_{bce}(\mathbf{x},\mathbf{y},\mathbf{w}) + \mu\mathscr{L}_{top}(\mathbf{x},\mathbf{y},\mathbf{w}) \tag{3.2}$$

with respect to $\mathbf{w}$. μ is a scalar weighing the relative influence of both terms. We set it so that the order of magnitude of both terms is comparable. Fig. 3.2(a) illustrates the proposed approach.

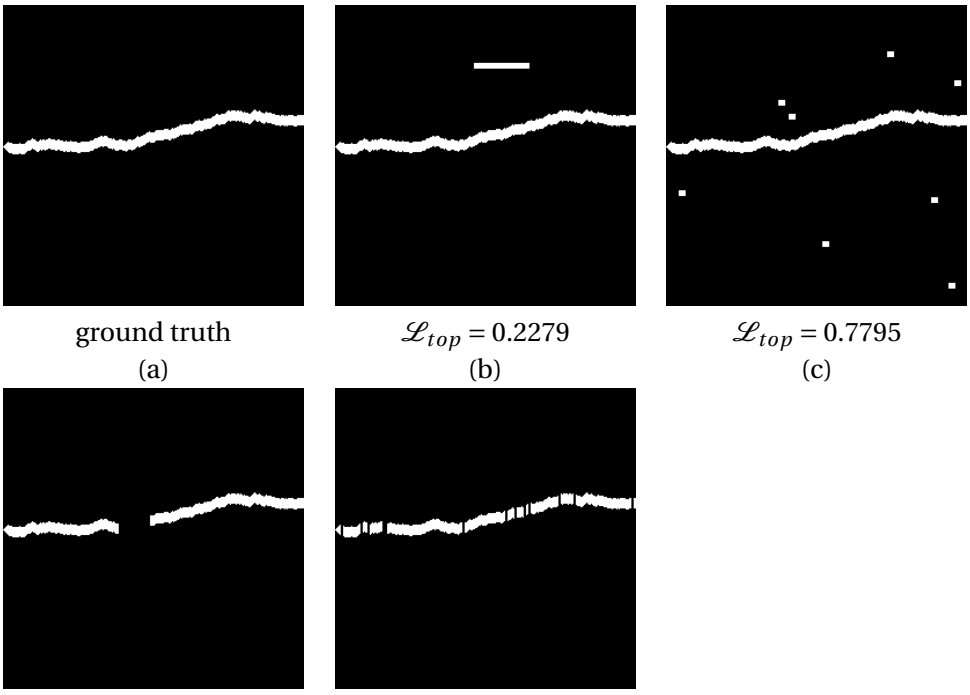


Figure 3.3 – **The effect of mistakes on topology loss.** (a) Ground truth (b)-(e) we flip 240 pixels in each prediction, so that $\mathscr{L}_{bce}$ is the same for all of them, but as we see $\mathscr{L}_{top}$ penalizes more the cases with more small mistakes, which considerably change the structure of the prediction.

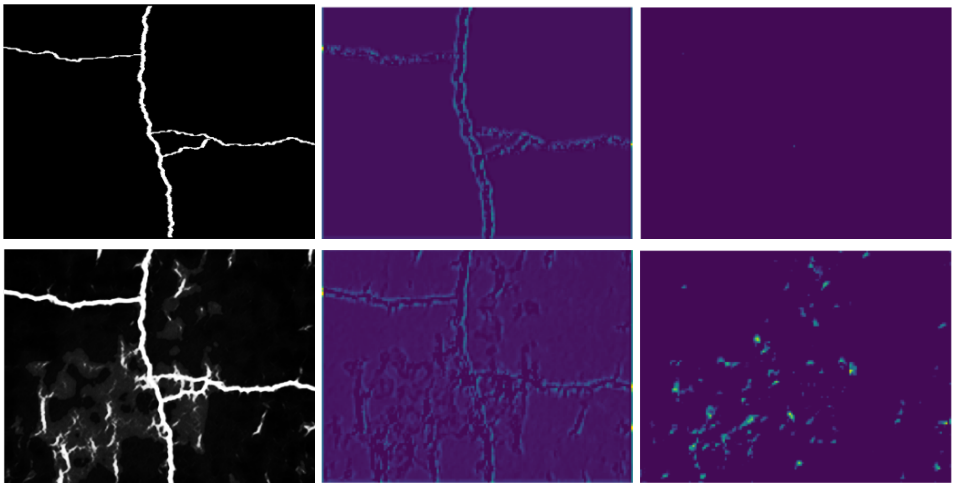


Figure 3.4 – **Activations in two example VGG layers.** (a) Ground-truth (top) and corresponding prediction with errors (bottom). (b) Responses of a VGG19 channel specialized in elongated structures. (c) Responses of a VGG19 channel specialized in small connected components. $\mathscr{L}_{top}$ strongly encourages responses in the former and penalizes responses in the latter.

3.2.3 Iterative refinement

The topology loss term of Eq. 3.1 improves the quality of the predictions. However, as we will see in Section 3.3, some mistakes still remain. They typically show up in the form of small gaps in lines that should be uninterrupted. We iteratively refine the predictions to eliminate such problems. At each iteration, the network takes both the input image and the prediction of the previous iteration to successively provide better predictions.

In earlier works that advocate a similarly iterative approach [78, 98], a different module f^k is trained for each iteration k, thus increasing the number of parameters of the model and making training more demanding in terms of the amount of required labeled data. An interesting property of this iterative approach is that the correct delineation $\mathbf{y}$ should be the fixed point of each module f^k, that is, feeding the correct delineation should return the input

$$\mathbf{y} = f^k(\mathbf{x} \oplus \mathbf{y}), \tag{3.3}$$

where $\oplus$ denotes channel concatenation and we omitted the weights of f^k for simplicity. Assuming that every module f^k is Lipschitz-continuous on $\mathbf{y}$,[2] we know that the fixed-point iteration

$$f^k(\mathbf{x} \oplus f^k(\mathbf{x} \oplus f^k(\ldots))) \tag{3.4}$$

converges to $\mathbf{y}$. We leverage this fixed-point property to remove the necessity of training a different module at each iteration. Instead, we use the same single network f at each step of the refinement pipeline, as depicted in Fig. 3.2(b). This makes the model much simpler and less demanding of labeled data for training.

This approach has also been applied before in [86] for image segmentation. Their application of the recurrent module was oriented towards increasing the spatial context. Rather than doing that, we keep the scale of the input to the modules fixed by zero-padding, in order to exploit the capacity of the network to correct its own errors. We show that it helps the network to learn a contraction map that successively improves the estimations. The predictive model can therefore be expressed as

$$\hat{\mathbf{y}}^{k+1} = f(\mathbf{x} \oplus \hat{\mathbf{y}}^k, \mathbf{w}), \quad k = 0, \ldots, K-1\,, \tag{3.5}$$

where K is the total number of iterations and $\hat{\mathbf{y}}^K$ the final prediction. We initialize the model with an empty prediction $\hat{\mathbf{y}}^0 = \mathbf{0}$.

[2] Lipschitz continuity is a direct consequence of the assumption that every f^k will always improve the prediction of the previous iteration.

Instead of minimizing only the loss for the final network output, we minimize a weighted sum of partial losses. The k-th partial model, with $k \leq K$, is the model obtained from iterating Eq. 3.5 k times. The k-th partial loss $\mathscr{L}^k$ is the loss from Eq. 3.2 evaluated for the k-th partial model. Using this notation, we define our *refinement* loss as a weighted sum of the partial losses

$$\mathscr{L}_{ref}(\mathbf{x},\mathbf{y},\mathbf{w}) = \frac{1}{Z}\sum_{k=1}^{K} k\,\mathscr{L}^k(\mathbf{x},\mathbf{y},\mathbf{w})\,, \tag{3.6}$$

with the normalization factor $Z = \sum_{k=1}^{K} k = \frac{1}{2}K(K+1)$. We weigh more the losses associated with the final iterations to boost the accuracy of the final result. However, accounting for the earlier losses enables the network to learn from all the mistakes it can make along the way and increases numerical stability. It also avoids having to preprocess the predictions before re-injecting them into the computation, as in [43].

In practice, we first train a single module network, that is, for $K = 1$. We then increment K, retrain, and iterate. We limit K to 3 during training and testing as the results do not change significantly for larger K values. We will show that this successfully fills in small gaps.

3.3 Results

Data. We evaluate our approach on three datasets featuring very different kinds of linear structures:

1. **Cracks:** Images of cracks in road [127]. It consists of 104 training and 20 test images. As can be seen in Fig. 3.7, the multiple shadows and cluttered background makes their detection a challenging task. Applications include quality inspection and material characterization.

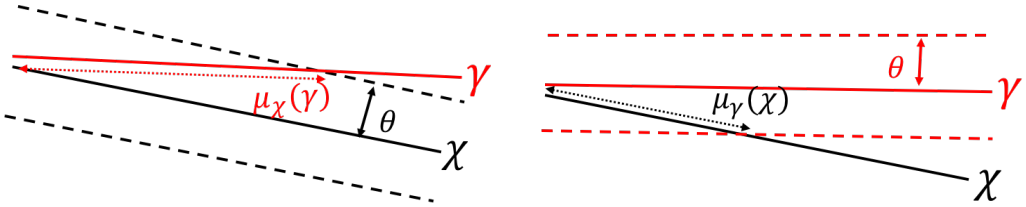
2. **Roads:** The Massachusetts Roads Dataset [69] is one of the largest publicly available collections of aerial road images, containing both urban and rural neighbourhoods, with many different kinds of roads ranging from small paths to highways. The set is split into 1108 training and 49 test images, 2 of which are shown in Fig. 3.6.

3. **EM:** We detect neuronal boundaries in Electron Microscopy images from the ISBI'12 challenge [6] (Fig. 3.8). There are 30 training images, with ground truth annotations, and 30 test images for which the ground-truth is withheld by the organizers. Following [96], we split the training set into 15 training and 15 test images. We report our results on this split.

Training protocol. Since the U-Net cannot handle very large images, we work with patches of 450×450 pixels for training, whoich are big enough to capture the context. We perform data augmentation mirroring and rotating the training images by 90°, 180° and 270°. Additionally, in the **EM** dataset, we also apply elastic deformations as suggested in [88] to compensate for the small amount of training data. Ground-truth of **Cracks** dataset consists of centerlines, so we dilate it by margin of 4 pixels to perform segmentation. We use batch normalization for faster convergence and use current batch statistics also at the test time as suggested in [16]. We chose Adam [49] with a learning rate of 10^{-4} as the optimization method.

Pixel-wise metrics. Our algorithm outputs a probabilty map, which lends itself to evaluation in terms of precision- and recall-based metrics, such as the F1 score [96] and the precision-recall break-even point [69]. They are well suited for benchmarking binary segmentations, but their local character is a drawback in the presence of thin structures. Shifting a prediction even by a small distance in a direction perpendicular to the structure yields zero precision and recall, while still reasonably representing the data. We therefore evaluate the results in terms of *correctness*, *completeness*, and *quality* [117]. They are metrics designed specifically for linear structures, which measure the similarity between predicted skeletons and ground truth-ones. They are more sensitive to precise locations or small width changes of the underlying structures. Potential shifts in centerline positions are handled by relaxing the notion of a true positive from being a precise coincidence of points to not exceeding a distance threshold. Correctness corresponds to relaxed precision, completeness to relaxed recall, and quality to intersection-over-union. In order to define them, denote the set of ground truth skeleton pixels by $\mathscr{X}$ and the set of prediction skeleton pixels by γ. We define the subset of prediction skeleton pixels that *match* the ground truth as $\mu_{\mathscr{X}}(\gamma)$. Similarly, the subset of ground truth skeleton matching prediction is $\mu_{\gamma}(\mathscr{X})$. The matching is defined in terms of a threshold θ on a distance $d()$ to the nearest point of the other set $\mu_B(A) = \{a \in A | \exists b \in B,\ d(a,b) < \theta\}$. The correctness, a rough equivalent of precision, is then defined as $\frac{|\mu_{\mathscr{X}}(\gamma)|}{|\gamma|}$. Similarly, $completeness = \frac{|\mu_{\gamma}(\mathscr{X})|}{|\mathscr{X}|}$ and $quality = \frac{|\mu_{\mathscr{X}}(\gamma)|}{|\gamma| - |\mu_{\gamma}(\mathscr{X})| + |\mathscr{X}|}$. A graphical representation of matching procedure is presented in Fig. 3.5. In our experiments we use a threshold of distance $d = 2$ pixels for roads and cracks, and 1 for the neuronal membranes.

Topology-based metrics. The pixel-wise metrics are oblivious of topological differences between the predicted and ground-truth networks. A more topology-oriented set of measures was proposed in [115]. It involves finding the shortest path between two randomly picked connected points in the predicted network and the equivalent path in the ground-truth network or vice-versa. If no equivalent path exists, the former is classified as *infeasible*. It is classified as *too-long*/*-short* if the length of the paths differ by more than 10%, and as *correct*



(a) (b)

Figure 3.5 – **Completness and correctness** (a) Matching ground truth with prediction skeleton. (b) Matching prediction with ground truth skeleton.

otherwise. In practice, we sample 200 paths per image, which is enough for the proportion of correct, infeasible, and too-long/-short paths to stabilize.

The organizers of the EM challenge use a performance metric called *foreground-restricted random score*, oriented at evaluating the preservation of separation between different cells. It measures the probability that two pixels belonging to the same cell in reality and in the predicted output. As shown in Fig. 4.1(c,d), this kind of metric is far more sensitive to topological perturbations than to pixel-wise errors.

3.3.1 Ablation Study

We start with performing ablation study of the method. To isolate individual contribution of the two main components of my approach, we compare its variants:

- **OURS-NoRef**, our approach with the topological loss of Eq. 3.2 but no refinement steps ($K = 1$).
- **OURS**, our complete, iterative method including the topological term and $K = 3$ refinement steps. It is trained using the refinement loss of Eq. 3.6 as explained in Section 3.2.3.

As can be seen in Table 3.1, using all three first layers of the VGG network to compute the

VGG layers	Quality
None	0.3924
layer 1	0.6408
layer 2	0.6427
layer 3	0.6974
layers 1,2,3	0.7446

Table 3.1 – Effect of the VGG layers. Quality scores for **OURS-NoRef** method when using different VGG layers to compute the topology loss of Eq. 3.1 on the **Cracks** dataset.

Number of iterations	Quality
OURS-NoRef	0.5580
OURS 1 iteration	0.5621
OURS 2 iterations	0.5709
OURS 3 iterations	0.5722
OURS 4 iterations	0.5727

Table 3.2 – Effect of the number of refinement steps. Quality scores for **OURS** method on the **EM** dataset as a function of the number of refinement iterations. **OURS-NoRef** included for comparison.

Feature extractor	F1 score
None	0.7952
AlexNet	0.8053
VGG19 with random initialization	0.8037
VGG19	0.8140
VGG16	0.8106
ResNet	0.8190

Table 3.3 – Effect of the architecture used as a feature extractor. F1 scores for **OURS-NoRef** method on the **EM** dataset when using different architectures to compute the topology loss of Eq.(3.1)

topology loss yields the best results. While varying the value of μ, weighting factor in Eq. 3.1, we can deduce from Table 3.4 that the best strategy is to set it such that it has got comparable magnitude to binary cross entropy (*i.e.* 0.01). As shown in Table 3.3, using different pretrained networks as topology feature extractors has positive effect on the results. Similarly, we evaluated the impact of the number of improvement iterations on the resulting performance on the **EM** dataset, which we present in Table 3.2. The performance stabilizes after the third iteration. We therefore used three refinement iterations in all further experiments. Note that in Table 3.2(right) the first iteration of **OURS** yields a result that is already better than **OURS-NoRef**. This shows that iterative training not only makes it possible to refine the results by iterating at test time, but also yields a better standalone classifier.

3.3.2 Comparison to Other Methods

We compare the results of our method to the following baselines:

- **CrackTree** [127] a crack detection method based on segmentation and subsequent graph construction. It consists of three steps: the first one removes the shadows visible on the pavement, at the same time preserving the cracks. In the second step a crack

μ parameter	F1 score
0	0.7952
0.001	0.8059
0.01	0.8142
0.1	0.8140
10	0.8058
inf	0.7987

Table 3.4 – Effect of the weighting topology loss and binary cross-entropy. F1 scores for **OURS-NoRef** method on the **EM** dataset when using different weighting parameter of Eq.(3.2).

probability map is produced using a tensor voting. Finally, crack seeds are sampled from the probability map and constitute vertices of a graph, from which a Minimum Spannoing Tree is found.

- **MNIH** [69], a fully-connected neural network for road segmentation using 64×64 image patches.
- **HED** [118], a nested, multi-scale approach for edge detection, which leverages fully-convolutional neural network to learn hierarchical representation of edges. Initially designed for edge tectection, its hierarchical structure helps resolve the inherent ambiguity associated with assigning edge boundaries. Similarly to our approach, it computes the training loss on multiple scales.
- **U-Net** [88], pixel labeling using the fully-convolutional U-Net architecture comprising of 4 downsampling and 4 upsampling steps, trained with BCE loss.
- **CHM-LDNN** [96], a multi-resolution recursive approach to delineating neuronal boundaries. It consists of two parts. Logic Disjunctive Normal Networks (LDNN) are one-layer feature detectors implemented as sigmoids and Cascaded Hierarchical Model (CHM) comprising of classifiers learnt at different scales, which recursively refine their predictions.
- **Reg-AC** [102], a regression-based approach to finding centerlines and refining the results using autocontext.

We reproduce the results for **HED**, **MNIH**, **U-Net** and **Reg-AC**, and report the results published in the original work for **CHM-LDNN**.

We report results of our comparative experiments for the three datasets in Tables 3.5, 3.6, 3.7, and 3.8. Even without refinement, the topological loss outperforms all the baselines. Refinement boosts the performance yet further. The differences are greater when using the metrics specifically designed to gauge the quality of linear structures in Table 3.7 and even more when

Method	P/R
MNIH	0.6822
HED	0.7107
U-Net	0.7460
OURS-NoRef	0.7610
OURS	**0.7782**

Table 3.5 – Experimental results on the **Roads** dataset. Precision-recall break-even point (P/R). Note the results are expressed in terms of the standard precision and recall, as opposed to the relaxed measures reported in [69].

Method	F1
CHM-LDNN	0.8072
HED	0.7850
U-Net	0.7952
OURS-NoRef	0.8140
OURS	**0.8230**

Table 3.6 – F1 scores of the **EM** dataset.

using the topology-based metrics in Table 3.8. This confirms the hypothesis that our contributions improve the quality of the predictions mainly in its topological aspect. The improvement in per-pixel measures, presented in Tables 3.5 and 3.6 suggests that the improved topology is correlated with better localisation of the predictions.

Finally, we submitted the results to the ISBI challenge server for the **EM** task. We received a foreground-restricted random score of 0.981. This puts our approach in first place among algorithms relying on a single classifier without additional processing. In second place is the recent method of [98], which achieves the slightly lower score of 0.978 even though it relies on a significantly more complex classifier.

3.3.3 Qualitative Results

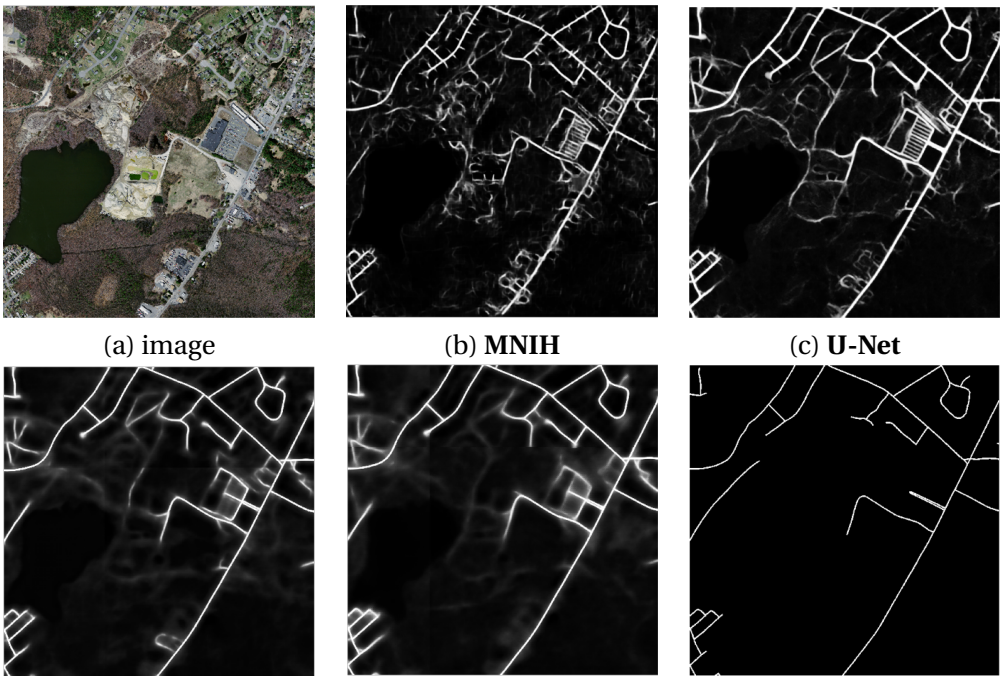
Figs. 3.7, 3.6, and 3.8 depict typical results on the three datasets. Note that adding the topology loss term and iteratively refining the delineations makes our predictions more structured and consistently eliminates false positives in the background, without losing the curvilinear structures of interest as shown in Fig. 3.9. For example, in the aerial images of Fig. 3.6, line-like structures such as roofs and rivers are filtered out because they are not part of the training data, while the roads are not only preserved but also enhanced by closing small gaps. In the case of neuronal membranes, the additional topology term eliminates false positives corresponding to cell-like structures such as mitochondria.

Dataset	Method	Correct.	Complet.	Quality
Cracks	**Reg-AC**	0.1070	0.9283	0.1061
	U-Net	0.4114	0.8936	0.3924
	OURS-NoRef	0.7955	0.9208	0.7446
	OURS	**0.8844**	**0.9513**	**0.8461**
Roads	**Reg-AC**	0.2537	0.3478	0.1719
	MNIH	0.5314	0.7517	0.4521
	U-Net [1]	0.6227	0.7506	0.5152
	OURS-NoRef	0.6782	0.7986	0.5719
	OURS	**0.7743**	**0.8057**	**0.6524**
EM	**Reg-AC**	0.7110	0.6647	0.5233
	U-Net [1]	0.6911	0.7128	0.5406
	OURS-NoRef	0.7096	0.7231	0.5580
	OURS	**0.7227**	**0.7358**	**0.5722**

Table 3.7 – Correctness, completeness and quality scores for extracted centerlines.

Dataset	Method	Correct	Infeasible	2Long 2Short
Cracks	**Reg-AC**	39.7	56.8	3.5
	U-Net	68.4	27.4	4.2
	OURS-NoRef	90.8	6.1	3.1
	OURS	**94.3**	3.1	2.6
Roads	**Reg-AC**	16.2	72.1	11.7
	MNIH	45.5	49.73	4.77
	U-Net [1]	56.3	38.0	5.7
	OURS-NoRef	63.4	32.3	4.3
	OURS	**69.1**	24.2	6.7
EM	**Reg-AC**	36.1	38.2	25.7
	U-Net	51.5	16.0	32.5
	OURS-NoRef	63.2	16.8	20.0
	OURS	**67.0**	15.5	17.5

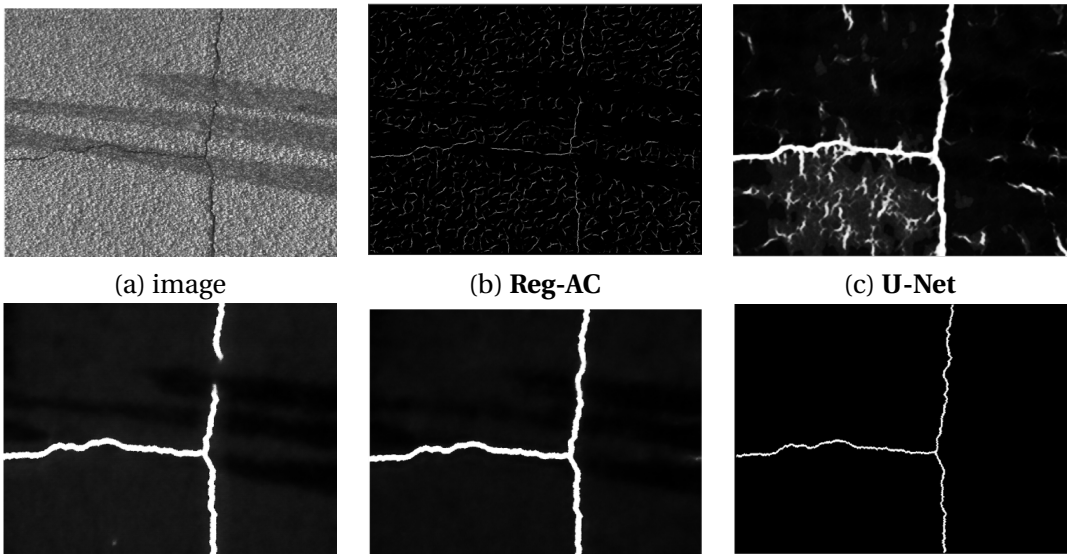
Table 3.8 – The percentage of correct, infeasible and too-long/too-short paths sampled from predictions and ground truth.



(a) image (b) **MNIH** (c) **U-Net**

(c) **OURS-NoRef** (d) **OURS** (e) ground-truth

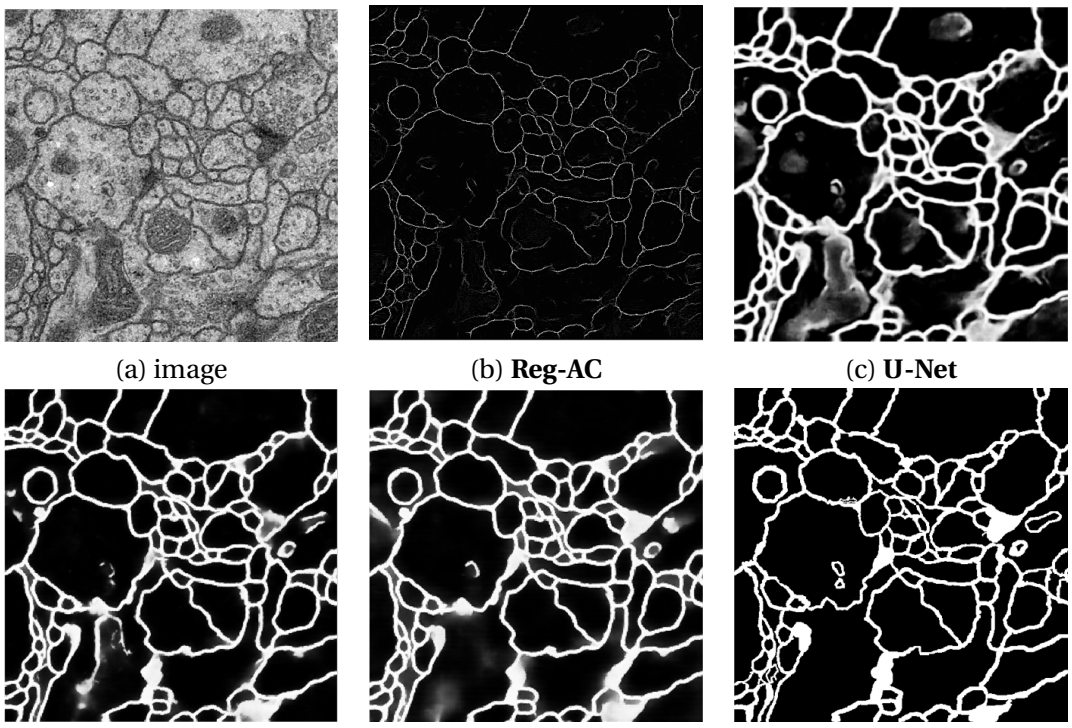
Figure 3.6 – **Roads:** Qualitative results.



(a) image (b) **Reg-AC** (c) **U-Net**

(c) **OURS-NoRef** (d) **OURS** (e) ground-truth

Figure 3.7 – **Cracks:** Qualitative results.



(a) image (b) **Reg-AC** (c) **U-Net**

(c) **OURS-NoRef** (d) **OURS** (e) ground-truth

Figure 3.8 – **Membranes:** Qualitative results.

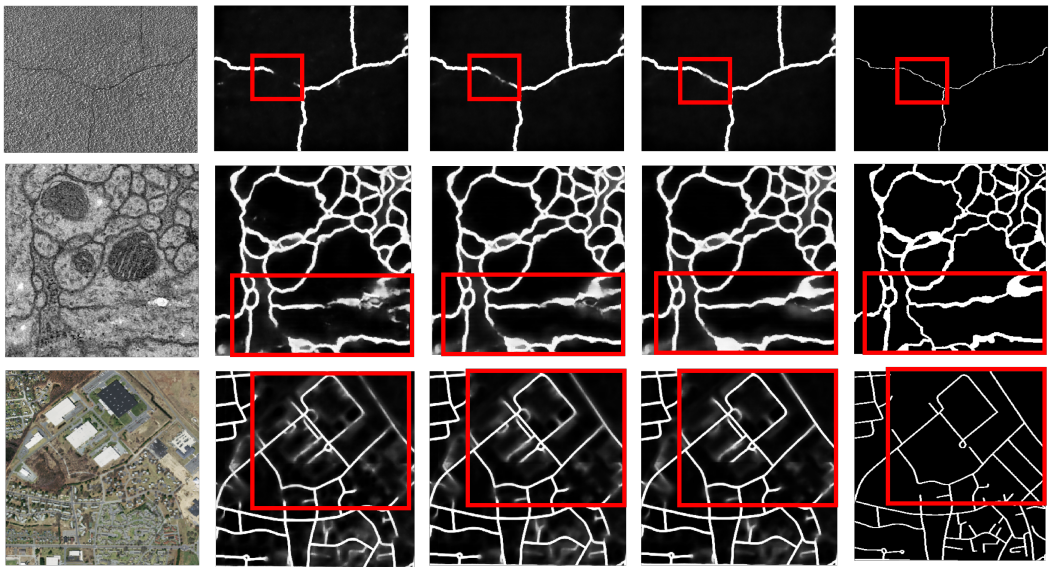


Figure 3.9 – **Iterative Refinement.** Prediction after 1, 2 and 3 refinement iterations. The right-most image is the ground-truth. The red boxes highlight parts of the image where refinement is closing gaps.

3.4 Conclusion

We have introduced a new loss term that accounts for topology of curvilinear structures by exploiting their higher-level features. We have further improved it by applying recursive refinement that does not increase the number of parameters to be learned. Our approach is generic and can be used for detection of many types of linear structures including roads and cracks in natural images and neuronal membranes in micrograms. We have relied on the U-Net to demonstrate it but it could be used in conjunction with any other network architecture.

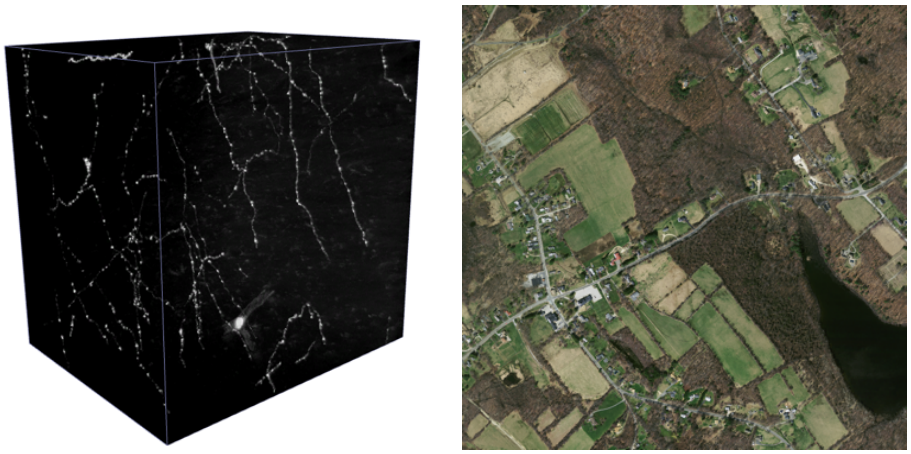
4 Multi-task Learning for Detection of Curvilinear Structures

One of the most challanging aspects in delineation is inferring the graph representation of the curvilinear network, especially important in neuroscience and cartography applications. Most of the existing delineation approaches first perform binary segmentation of the image and then refine it using either a set of hand-designed heuristics or applying a separate classifier that assigns likelihood to paths extracted from the pixel-wise prediction. In our work, we bridge the gap between segmentation and path classification, by training a deep network that performs those two tasks simultanously. We show that this approach is beneficial for the reconstruction, as it enforces consistency in the processing pipeline. We apply our method on two very different datasets depicting axons in microscopic images and roads in satellite imagery.

4.1 Introduction

Automated delineation of curvilinear structures, such as those shown in Fig. 4.1, has been investigated since the inception of Computer Vision in the 1960s and 1970s. Nevertheless, despite decades of sustained effort, full automation remains elusive when the image data is noisy and the structures are complex. As in many other fields, the advent of Machine Learning techniques in general, and Deep Learning in particular, has produced substantial advances, in large part because learning features from the data makes them more robust to appearance variations [30, 71, 102, 115, 109].

However, there remains a disconnect between two broad classes of techniques. Some attempt to produce *segmentation masks* in which pixels potentially belonging to structures of interest are labeled as foreground and others as background. Others take such masks, usually along with a scalar image that indicates how confident the system is about the assigned labels, as input that attempts to produce a *delineation*, that is, a graph-like representation of the linear



(a) (b)

Figure 4.1 – **Curvilinear structures.** (a) Axons and dendrites in a 2-photon light-microscopy stack. (b) Road network in aerial image.

structures. Our own earlier work [109] is representative of this problem: We use one kind of classifier [102] to produce the segmentation masks and another to score potential edges in the graph of which the desired delineation should be a subgraph. Similarly, more recent Deep Learning-based approaches [67, 112] rely on two indepedently-trained classifiers, one to perform segmentation and the other classify potential connections.

In this chapter, we bridge the gap between these two key steps so that they can be jointly optimized. To this end, we train a neural network made of an encoder and two different decoders to perform two separate tasks; first, given only an image, produce a segmentation mask. Second, given a path between two points as an additional input, return the likelihood that this path corresponds to a linear structure really present in that image. This enables us to compute a segmentation, build a graph whose edges correspond to candidate linear structures, weight them, and only retain only the best ones. This is a much streamlined version of our earlier approach [109]. Yet, because it enforces consistency throughout the processing chain, it performs better.

Our contribution is therefore a unified approach to segmenting linear structures and classifying linear paths. The intuition behind it is that these two tasks are closely related and should rely on the same features, specifically those that the decoder part of our neural network produces. This results in a general-purpose approach. We will demonstrate that it outperforms state-of-the-art ones in a number of image modalities, including including satellite and light microscopy images.

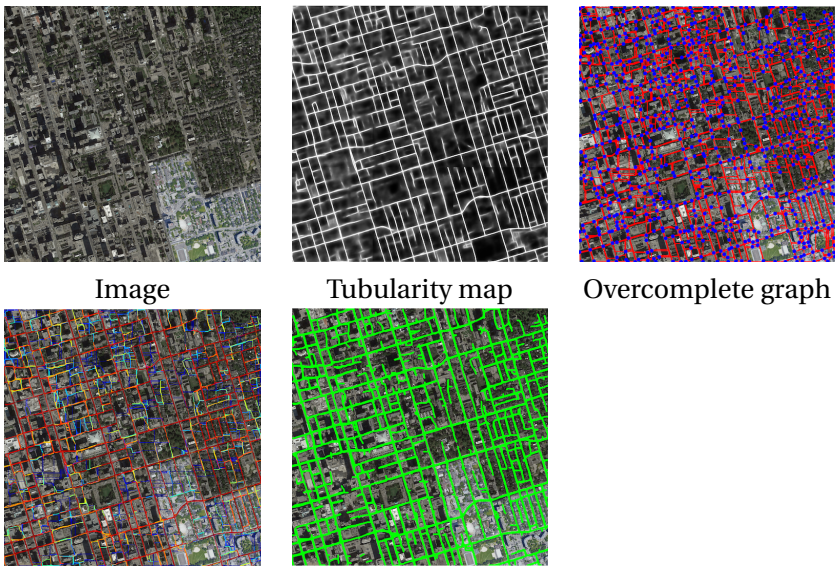


Figure 4.2 – **Delineation pipeline.** Given an image, we first compute pixel probability of belonging to the structure of interest, resulting in a tubularity map. Next, we build a graph by finding nodes, shown as blue dots, and connecting them by shortest paths, shown in red. This graph is overcomplete in the sense that the true paths are expected to correspond to a subset of its edges. To eliminate the ones that do not correspond to true linear structures, we run a classifier and remove the low-scoring ones.

4.2 Method

Most existing approaches to graph-based delineation comprise three steps:

1. Use one of the segmentation techniques to assign a pixel or voxel a probability of belonging to a linear or tubular structure, which we will refer to as a *tubularity map*. For example, in our earlier work, we used either OOF [55, 108] or a decision-based tree classifier [102].

2. Define a graph whose edges connect spatial locations and correspond to potential fragments of the target structures, for example by finding shortest paths between image locations where the distances take into account the pixel values of tubularity map [11, 72, 116, 109].

3. Find a subgraph within that graph whose edges correspond to the candidate target structure fragments that are retained in the final delineation. The simplest way to do this is to look for a minimum spanning tree [111, 84, 119], which can then be pruned. A more sophisticated approach that makes it possible to impose global geometric and topological constraints on the final delineation is to formulate the search for the

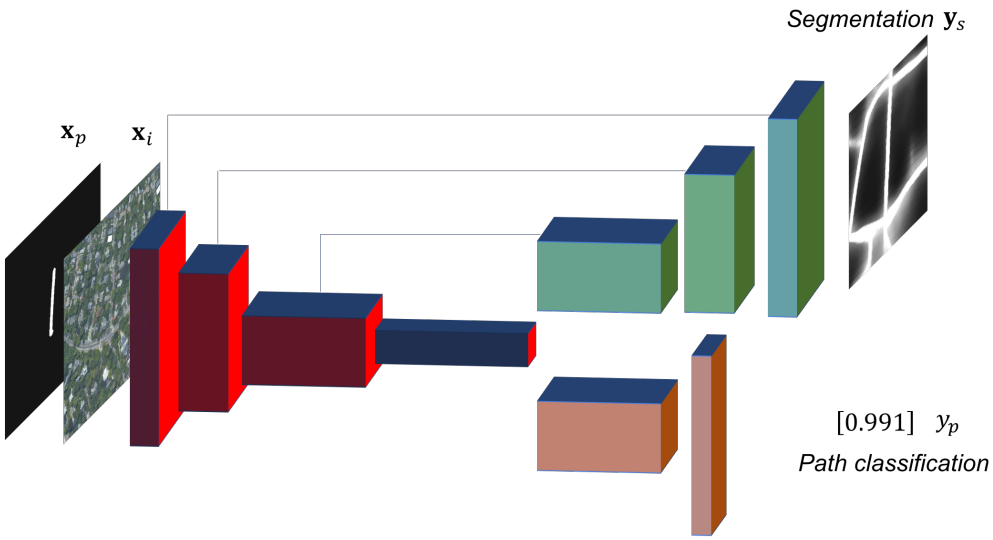


Figure 4.3 – **Two-stream architecture.** Both branches share the same U-Net encoder that takes as input the image and a binary mask representing a candidate path. The first branch uses the U-Net decoder and skip connections to produce a tubularity map. The second branch relies on a simpler network to yield a classification score for the path.

optimal subgraph as Linear or Quadratic program [109, 75], which allows reconstructing also loopy graphs. A recent trend is to also use two separate deep networks for this purpose. One is trained to perform the segmentation and the other to score proposed paths [67, 125, 112] or to trace the structure pixel by pixel and detect end points [8].

Our approach belongs in the same class. We compute a tubularity map, use it to build an overcomplete graph whose edges correspond to candidate linear structures, assign each one a score, and only retain the best ones. Fig. 4.2 depicts these steps. Unlike in [109], we did not find it necessary to perform a sophisticated linear and quadratic mixed-integer optimization to select an optimal subgraph. This is because we use the same network to compute the tubularity map and the edge weights, which ensures consistency and increases performance to the point where the more sophisticated approach to building the final delineationis no longer needed to outperform the state-of-the-art. In this section we describe the architecture of this network, along with the associated training and testing procedures.

4.2.1 Formalization

For simplicity's sake, we formalize our approach in the context of 2D image delineation but it naturally extends to 3D image stacks.

Let $\mathbf{x}_i \in \mathbb{R}^{H \cdot W \cdot C}$ be a $W \times H$ C-channel image and $\mathbf{x}_p \in \mathbb{R}^{H \cdot W}$ be a $W \times H$ binary image repre-

senting an overlaid candidate path, where 1s denote the pixels belonging to the path. They are inputs to our network f and are depicted on the left side of Fig. 4.3. As shown on the right side of the figure, the network outputs $f(\mathbf{x}_i, \mathbf{x}_p)$ are a floating point segmentation image $\mathbf{y}_s \in \{0,1\}^{H \cdot W}$ and classification score y_p for the path.

Every pixel value of $\mathbf{y}_s$ is understood to be the probability of pixel i having label 1, that is, of belonging to a linear structure. Given the ground-truth $W \times H$ binary segmentation image $\mathbf{y}_s^{gt}$, we therefore want the segmentation training loss L_{seg}, the binary cross entropy summed over $\mathbf{y}_s$ and $\mathbf{y}_s^{gt}$, to be as small as possible. Similarly the path classification score y_p is taken to be the probability of the path being valid and we also want binary cross entropy $\mathscr{L}_p$ given the ground-truth label y_p^{gt} to also be as small as possible.

Therefore for each training example $\mathbf{x}_i, \mathbf{x}_p, \mathbf{y}_s, y_p$, we take the loss to be

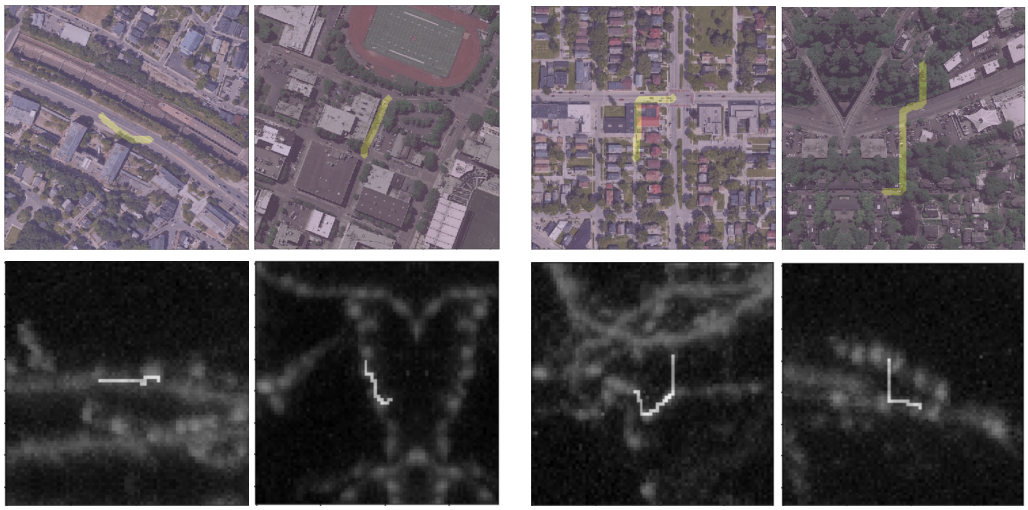
$$\mathscr{L}(\mathbf{x}_i, \mathbf{x}_p, \mathbf{y}_s, \mathbf{y}_p; \mathbf{w}) = \mathscr{L}_{seg} + \eta_p \mathscr{L}_p \,, \tag{4.1}$$

where $\mathbf{w}$ represents the network weights and η_p is a constant that ensures that the gradients associated to both losses have similar magnitudes. Optionally, we can add a topology loss presented in Chapter 3 to $\mathscr{L}$ to ensure that the global statistical properties of the segmentation are similar to those of ground-truth.

4.2.2 Architecture

To implement the network f described above, we start from the U-Net architecture [88]. It is fully convolutional and comprises an encoder and decoder with skip connections. It has been shown to be state-of-the-art for binary segmentation of linear structures [16, 73]. We therefore modified it to accept as input the binary mask $\mathbf{x}_p$ in addition to the image $\mathbf{x}_i$, as shown on the left side of Fig. 4.3. We then add a second branch depicted at the bottom of the figure, that connects to the encoder and outputs the classification score y_p while the first branch still produces the segmentation probability map $\mathbf{y}_s$. Both branches of modified two-stream U-Net are depicted on the right side of Fig. 4.3.

More precisely the segmentation branch follows the standard U-Net design with 4 max pooling operations and skip connections between corresponding encoder and decoder layers. At each downsampling step the number of filters is increased by a factor two and the reverse occurs while upsampling. The path classification branch uses the same encoder but a much simpler decoder that comprises a convolutional layer followed by max pooling and two fully connected layers. To speed-up convergence at training time, we use batch normalization [16]. In case of 3D stacks we use the decoder with 3 max-pooling operations and one fully-connected layers due to memory constraints.



Positive samples Negative samples

Figure 4.4 – **Candidate road and neuron paths. (Top row)** Road images with candidates overlaid in yellow. **(Bottom row)** 2-Photon images with candidates overlaid in white. In both cases, we treat the paths that remain with linear structures as positive, even if they do not exactly follow the centerline. By contrast, paths that cross from one structure to another or take shortcuts as treated as negative.

In practice, to compute segmentation probability map $\mathbf{y}_s$ and path classification score y_p, we perform a first pass through the network with $\mathbf{x}_p$ replaced by an image entirely made of zeros, so that the result is only conditioned on the image. We then make a second pass using $\mathbf{x}_p$ to obtain a value of y_p that is conditioned both on the image and the input path. This guarantees that the same intermediate features are used to encode both the segmentation map and path classification score.

4.2.3 Training

To train our network, we need pairs of images $(\mathbf{x}_i, \mathbf{y}_s^{gt})$ and candidate paths x_p that are labeled as being either valid or not. As will be discussed in Section 4.3, there are publicly available training databases with ground truth segmentations we can use to obtain sufficiently large numbers of $\mathbf{x}_i$ and $\mathbf{x}_s^{gt}$. Given those, a simple way to find path candidates would be to sample positives from the ground-truth segmentation and use random paths as negatives. However, this would produce candidates that are far too easy to classify to provide a useful supervisory signal. Instead, we initially train only the segmentation branch of our network. When the loss stabilizes, we compute a tubularity image for each training image from which we sample positive and negative path samples. This enables us jointly train the two branches of the network. This way, the paths we train the path-scoring branch of our network are realistic ones that resemble those we are likely to encounter at inference time, such as those of Fig. 4.4.

For example, note that the true positives do not have to exactly follow the centerline as long as they remain along the ground-truth path.

Given a tubularity image, our path generating procedure proceeds through the following three steps:

1. **Finding graph nodes.** We skeletonize the tubularity image and find *significant points*, that is, intersections and endpoints. Endpoints have a single neighbor while intersections have more than two. We ensure that the sampled significant points are not closer than small radius ϵ As these significant points may be unevenly spaced in the image, we also sample the skeleton at regular intervals d to limit the maximum distance between neighboring samples.

2. **Connecting graph nodes.** Fig. 4.5 depicts this node creation process, which guarantees that nodes that ought to be connected are within a distance d of each other. To account for small inaccuracies, we connect all pairs of nodes whose distance is less than $1.5d$. This results in an overcomplete graph such as the one depicted by Fig. 4.2 in which each edge corresponds to a shortest path between two nodes. We use the A* algorithm to extract these paths. Given a start and a target node, it iteratively grows the path by adding to it the pixel (x, y) that is a neighbor of the current path endpoint and minimizes

$$f(x,y) = c(x,y) + h(x,y)\,, \tag{4.2}$$
$$c(x,y) = 1.1 - p(x,y)\,, \tag{4.3}$$
$$h(x,y) = 0.5 * d(x,y)\,, \tag{4.4}$$

   where p is the tubularity value returned by our classifier and d is the Euclidean distance to the target. In effect, c is the cost of adding a new pixel to the path and h approximates the cost of the shortest path from (x, y) to the target. We found this to deliver a good comprise between computation speed on potentially very large graphs and close approximation of the true shortest path.

3. **Selecting positive and negative samples.** Paths that overlap with segmentation ground truth by more than 90% are considered as positive. The others are assigned a negative label. This yields positive and negative samples as those depicted by Fig. 4.4. It treats as positive the paths that remain within the linear structures, even if they do not exactly follow its centerline. By contrast, those that cross from one structure to another using a shortcut are labeled as negative. In earlier work [109], we found that the latter were one of the main sources of errors in the final network topology and that it was crucial to eliminate them.

To keep the notations simple in Section 4.2.1, we formulated our approach in terms of whole images. In practice, our network operates on patches that are cropped from the training images. To select such patches for training purposes, we use the positive and negative paths selected as discussed above and crop the image around them. If a path does not fit within the receptive field of our U-Net encoder, we split it.

4.2.4 Inference

Inference proceeds in a way that closely mirrors training, as shown in Fig. 4.2. We use the segmentation branch of our now trained two-stream U-Net to compute the tubularity map. We then skeletonize it and create the graph nodes using the procedure depicted by Fig. 4.5. We connect them using the same A* algorithm as before and score the resulting paths using the second branch of our two-stream U-Net. Finally, we retain only the highest scoring ones.

The key to the effectiveness of this procedure is that we use the *same* network to compute the tubularity map and to score the paths, which guarantees that both operations rely on the same image features.

4.3 Results

4.3.1 Datasets and Baselines

We evaluate our approach on two datasets depicting very different curvilinear structures:

- **Roads** [8]: One of the biggest publicly available road network datasets, featuring highways, urban roads, and rural paths. The training set comprises images of 25 cities and the training set 15 *other* cities. The images come from Google Maps, are of size 4096× 4096, and each cover 40 km^2. The ground-truth was obtained from Open Street Maps and may therefore be noisy. Since the training and testing sets come from different cities, the database is well-suited to assess generalization abilities.
- **Axons**: 3D stacks of two-photon laser scanning microscopy images depicting axons in mouse brains at different stages of their development. We use 8 volumes of size ranging from 217×206×54 to 322×277×136 for training and two other volumes of size 250×250×200 and resolution 0.27 × 0.27 × 0.9 µm for testing.

We will refer to our method as **OURS** and compare it to two state-of-the-art graph-based approaches for road and neuron delineation:

- **RoadTracer** [8]: a recent method for road detection based on a Convolutional Neural

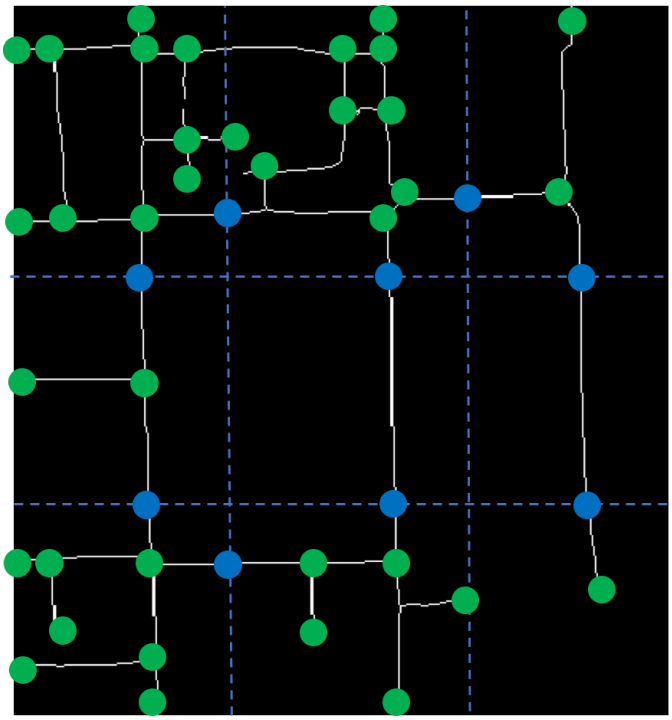


Figure 4.5 – **Creating graph nodes.** We skeletonize the tubularity map and find the topologically significant points, that is, the intersections and end-points shown as green disks. We use them as nodes of our over-complete graph. As they can be far from each other, we introduce the regular grid depicted by the dashed lines and use its intersections with the tubularity skeleton, shown as blue disks, as additional nodes. To prevent nodes from being too close from each other, we introduce an exclusion zone of radius ε, which is also the radius of the green disks, in which no additional nodes can be added. This approach guarantees that the distance between nodes that ought to be connected is always between ε and d.

Network trained to decide the direction of the next pixel on the traced path. We use the results computed by the authors on **Roads** dataset for evaluation.

- **MIP** [109]: Our previous approach to delineation that follows similar steps as **OURS**, but requires two separate classifiers, one to compute tubularity and the other to classify paths. It additionally performs Mixed Integer Programming optimization to find the optimal graph given the classified edges.

We also compare our results to those obtained by simply skeletonizing the segmentation, which we will refer to as **Seg**.

4.3.2 Evaluation Metrics

Our algorithm aims to produce paths that approximate the centerline of the linear structures while preserving the overall topology of the network. We therefore rely on topology-aware metrics that have been proposed in the literature to assess the performance of delineation algorithms. They all rely on computing and comparing the shortest path between a pairs of intersections or endpoints, which we defined as topologically significant points in Section 4.2.3, in the ground-truth and the shortest path between the corresponding significant points in the predicted graph. The similarity between the two paths is measured and used as a metric.

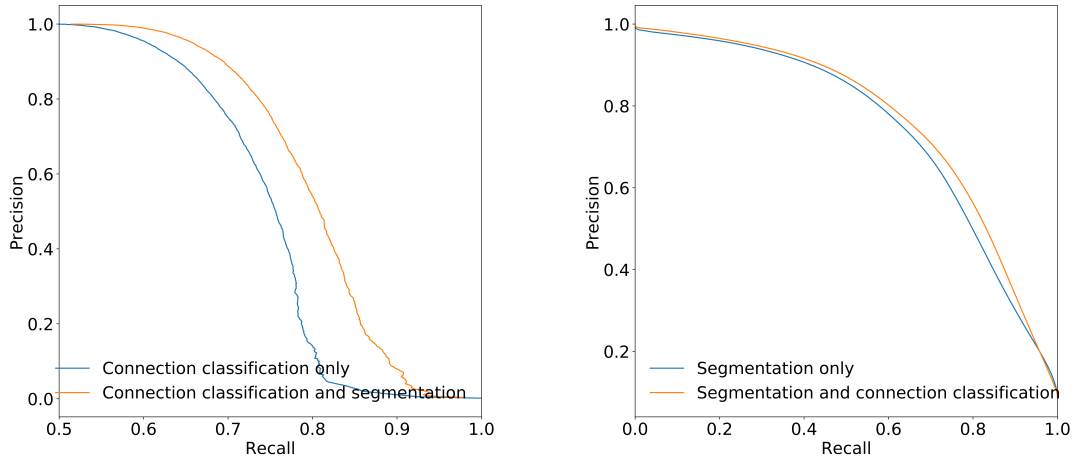
Normalized Path Difference [25]. Let a^* and b^* be any two significant points being connected by a ground-truth shortest-path of length l^*. If there are corresponding significant points a and b within a radius R in the prediction, we pick the closest ones, find the length l of the shortest-path connecting them, and compute a dissimilarity score

$$\min\{\frac{|l-l^*|}{l^*}, 1\}\,. \tag{4.5}$$

We then plot the histogram of Normalized Path Difference for all such testing paths. If there are no corresponding points in the prediction or the corresponding points are not connected in the predictions, the score is taken to be one. In practice we take the value of the radius R to be 40 pixels for **Roads** and 10 pixels in **Axons**, which is roughly the minimum spacing between two intersections.

Topological precision and recall [67]. For all pairs of significant points that are direct neighbors in the ground-truth graph, we consider the path connecting them and the corresponding shortest-path in the predicted graph, as described above. We can then write

$$\text{precision} = \frac{1}{\sum_p n_m} \sum_p \frac{n_m}{n_t} n_m \text{ and recall} = \frac{\sum_p n_m^*}{\sum_p n_t^*}\,, \tag{4.6}$$



(a) Path classification

(b) Pixel-wise segmentation

Figure 4.6 – **Task sharing.** Learning both path classification and segmentation simultaneously significantly boosts path classification (a) and does not harm segmentation (b), compared to training two networks seperately.

where n_m is the number of pixels along the predicted path that are within distance m of the ground-truth one, n_t is the total number of pixels in the predicted path. We sum them over all paths p in all images. Moreover, precision is the weighted sum of the proportion of pixels that are close enough, where the weights are the path lengths. Similarly, n_m^* is the number of pixels along the ground-truth path that are within distance m of the predicted one and n_t^* is the total number of pixels in the ground-truth path. When there is no corresponding path, n_m^* is set to zero and n_m is not counted to overall precision. Precision captures how accurate the predicted path locations are and recall the overall proportion of the ground truth that is effectively modeled by these paths.

4.3.3 Implementation

To train our network, we used Adam optimizer with learning rate 10^{-4}. As the road images have very high resolution, we reduced it by the order of two for training and testing. The results of **RoadTracer** method were computed for full resolution images, so in order to keep the fair comparison, we upsampled our testing results to the original size before computing the metrics.

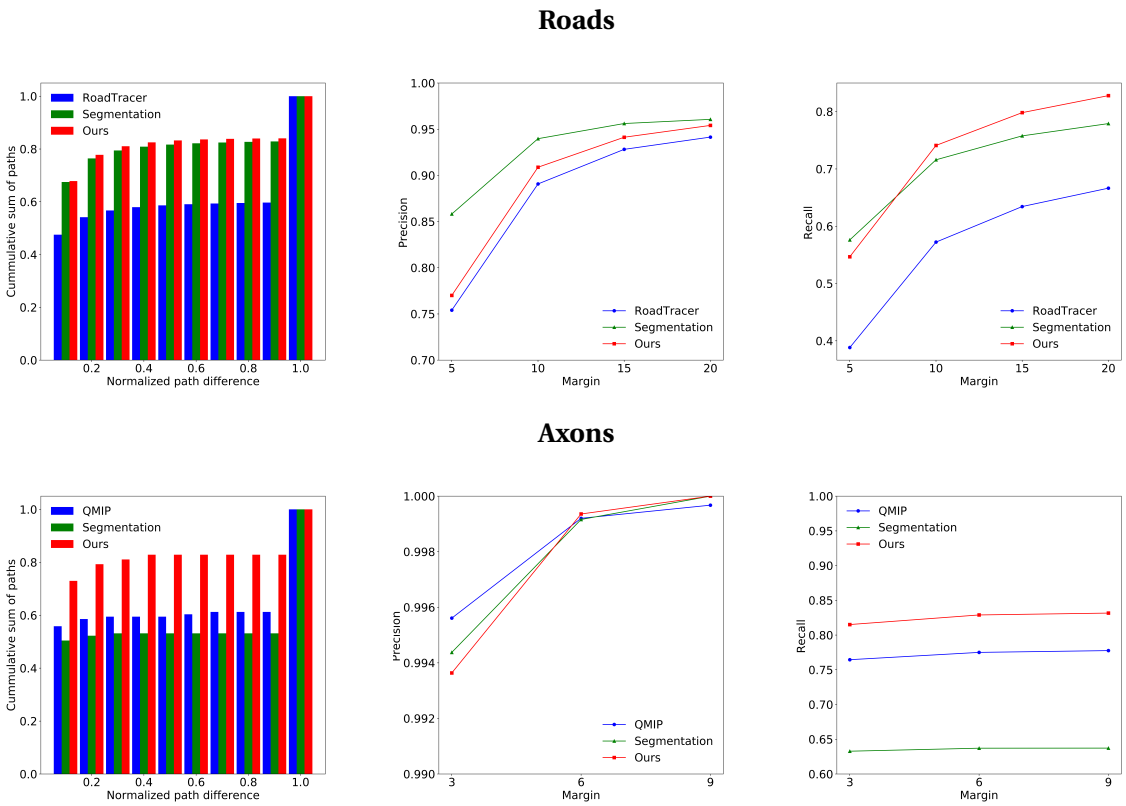


Figure 4.7 – **Quantitative results. Top:** Roads. **Bottom:** Axons. From left to right, cummulative distribution of the normalized path distances; topological precision and recall as a function of the margin threshold m (in pixels) used to compute them.

4.3.4 Analysis

We first studied the effect of joint segmentation and path classification training on both of the tasks. While we found that it does not affect pixel-wise segmentation significantly, it considerably boosts the performance of path classification evaluated on the road validation set, as shown in Fig 4.6. This is likely because both tasks share similar features, which emphasize the dependance of path classification on segmentation and help with distinguishing negative paths, even if they partly coincide with a true road.

4.3.5 Comparative Results

Fig. 4.8 and Fig. 4.9 depict some of our results on the **Roads** and **Axons** datasets. In Figs. 4.7, we report the corresponding quantitative results and compare them against those of our baselines. All three road extraction methods have very similar precision, but our approach has got much higher recall. The cummulative distribution of normalized path differences also

indicates that joint path classification and segmentation reduces the number of disconnected segments, as larger proportion of paths has got smaller length difference.

The qualitative results reveal that while our method produces some false positives, they are usually in ambiguous places such as parking lots or railway tracks (in case of roads) or faint axons.

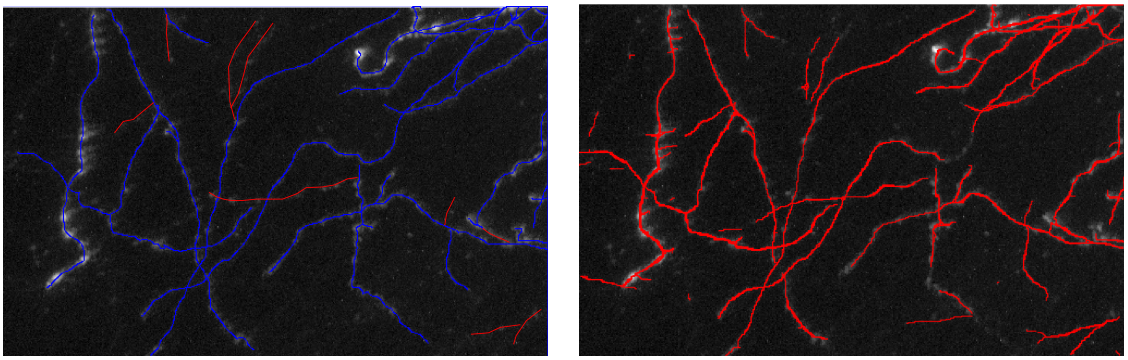
4.4 Conclusion

We presented a method for joint segmentation and connection classification of curvilinear structures. We use the fact that those two tasks share similarties to train a network that can perform both tasks simultanously. Our approach outperforms state-of-the-art methods on datasets depicting roads and neurons.

(a)

(b)

Figure 4.8 – **Roads qualitative results.** Kansas City road network (a) **RoadTracer** (b) **OURS**. True positive paths are shown in green, false positives in red and false negatives in blue.



(a) (b)

Figure 4.9 – **Axons qualitative results.** (a) ground-truth and (b) **OURS** reconstruction of one of the **Axons** test images. Note that the blue traces in ground-truth were supplied by the annotator and they were used to compute the metrics. The traces in red correspond to axons that are still visible in the background or were omitted by the annotator and the predictions corresponding to them should not be necessairly treated as false positives.

5 Active Learning Based on Local Consistency of Delineation

In the previous chapters we presented ways for segmenting and obtaining final graph reconstruction of linear networks. Their increased effectiveness, just like many other approaches, owes much to Machine Learning. However, it requires significant amounts of *annotated* data for training, which is tedious and time-consuming to obtain, especially in case of structures appearing in 3D images.

In this chapter, we describe an Active Learning (AL) approach that considerably speeds up the annotation process. Unlike standard methods, it takes advantage of the specificities of the delineation problem. It operates on a graph and can reduce the training set size by up to 80 % without compromising the reconstruction quality.

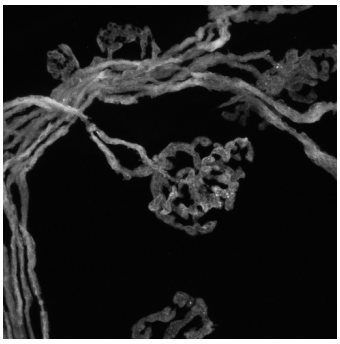
We will show that our approach outperforms conventional ones on various biomedical and natural image datasets, thus showing that it is broadly applicable.

Part of the material presented in this Chapter appeared in [74].

5.1 Introduction

Complex curvilinear structures present very different appearances and it has recently been shown that training classifiers to assess whether an image path is likely to be a structure of interest is key to improving the performance of automated delineation algorithms [111, 110, 11, 90, 72, 116].

However, while such Machine-Learning based algorithms are effective, they still require significant amounts of manual annotation for training purposes. For everyday scenes, this can be done by crowd-sourcing [60, 57]. In more specialized areas such as neuroscience or medicine, this is impractical because only experts whose time is scarce and precious can do this reliably. This problem is particularly acute when dealing with 3D image stacks, which are



(a)

(b)

Figure 5.1 – **Linear structures.** Images of two different neural structures obtained using confocal microscopy. The enormous variety of curvilinear structures requires problem-specific training datasets even in case of the same modality.

much more difficult to interact with than regular 2D images and require special expertise. It is further compounded by the fact that data preparation processes tend to be complicated and not easily repeatable leading the curvilinear structures to exhibit very different appearances as shown in Fig. 5.1. This means that a classifier trained on one acquisition will not perform very well on a new one, even when using the same modality.

In this chapter, we propose an Active Learning (AL) approach that exploits the specificities of delineation algorithms to massively reduce the effort and drudgery involved in collecting sufficient amounts of training data. At the heart of all AL methods is a query mechanism that enables the system to ask a user to label a few well chosen data samples, which it has selected based on how informative the answers are likely to be. AL has been successfully deployed in areas such as Natural Language Processing [107], Computer Vision [48], and Bioinformatics [58]. While it has made it possible to train classifiers with less of human intervention, none of the algorithms can exploit the fact that, for delineation purposes, the paths to be annotated form a graph in which neighborhood and geometric relationships can and should be considered.

In our approach, we explicitly use these relationships to derive multi-sample entropy estimates, which are better surrogates of informativeness than the entropy of individual samples that is typically used [20]. As a result, our queries focus more effectively on ambiguous regions in image space, that is, those at the boundary between positive and negative examples.

To avoid having to retrain the system after each individual query and further increase efficiency, we also integrate into our approach a batch strategy that lets the system ask the user several questions simultaneously. It incorporates *density* measures that ensure that the batches are diverse in features, representative of the delineation problem at hand and located near each

other in the images so as to facilitate the interaction. This is particularly important in 3D volumes where scrolling from one region to another far away is cumbersome and potentially confusing for the user.

In short, our contribution is an AL approach that is tailored for the delineation of complex linear structures. In that sense, it is specialized. However, it is also generic in the sense that it can handle a wide variety of different structures. We will show that it outperforms more traditional approaches on both 2- and 3-D datasets representing different kinds of linear structures, that is, roads, blood vessels, and neural structures.

5.2 Active Learning for Delineation

Graph-based network reconstruction algorithms have recently shown superior performance compared to methods based on segmentation. They not only recover the geometry of the problem, but also the correct connectivity, which is crucial in applications such as neuroscience [90, 116, 72, 111, 110, 77]. They largely owe their performance to supervised Machine Learning techniques that allow them to recognize promising linear paths.

These methods usually start by computing a tubularity measure, which quantifies the likelihood that a tubular structure exists at given image location. Next, a set of subsampled high-tubularity superpixels [90, 116, 72] or longer paths [111, 110, 11, 77] are extracted. Each of them can be considered as an edge e_i belonging to overcomplete spatial graph $\mathcal{G}$ and characterized by a feature vector $\mathbf{x}_i$. Given two possible class labels ($y_i = 1$) and ($y_i = 0$) , a discriminative classifier assigns to each edge e_i probability of belonging to the structure of interest $p(y_i = 1|\mathbf{x}_i)$ or to the background, $p(y_i = 0|\mathbf{x}_i)$.

The optimal subgraph $\mathcal{T}^*$ can then be taken to be tree that minimizes the cost function over all trees $\mathcal{T}$ that are subgraphs of $\mathcal{G}$

$$\sum_{e_i \in E_{\mathcal{T}}} -\log \frac{p(y_i = 1|\mathbf{x}_i)}{p(y_i = 0|\mathbf{x}_i)}, \tag{5.1}$$

where $E_{\mathcal{T}}$ represents the edges of $\mathcal{T}$. Provided that one does not take into account the geometry of the tree but only its topology, this can be shown to be Maximum a Posteriori estimate. In practice, however, it is more effective to formulate the MAP problem in terms of pairs of consecutive edges. This makes it possible to introduce better geometric constraints [111] and to find generic subgraphs as opposed to only trees [110].

Whether using single edges or pairs, the key requirement for this kind of approach to perform

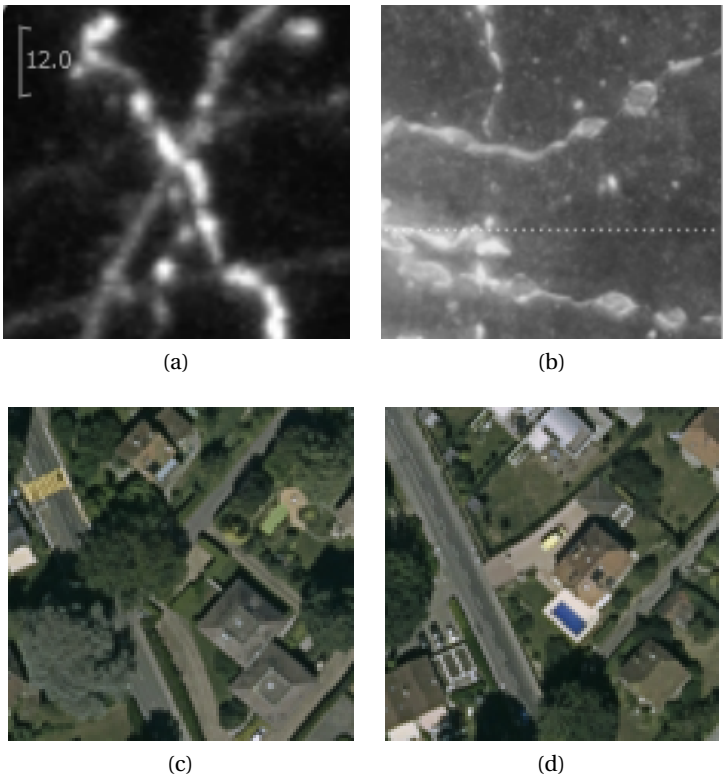


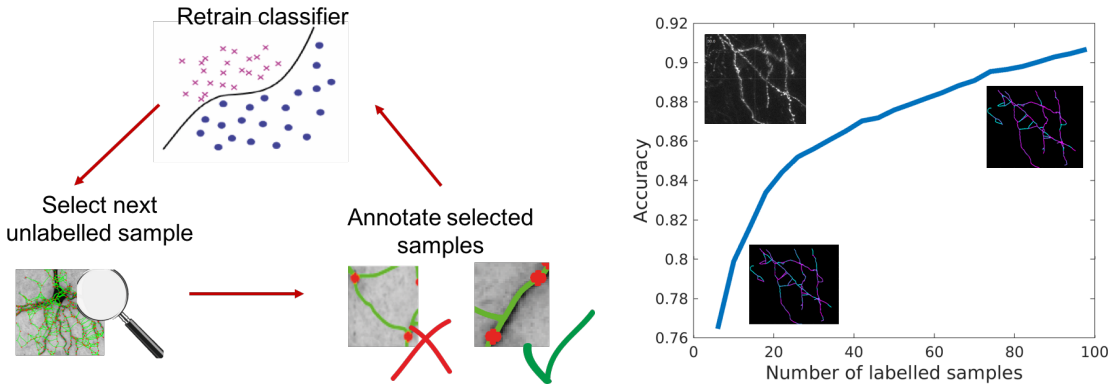
Figure 5.2 – **Ambiguous image regions.** (a) Branch intersection. (b) Discontinuities due to uneven tissue staining. (c) Discontinuities due to occlusion by a tree. (d) Linear structures such as driveways that should be ignored.

well is that the classifiers used to estimate the probabilities of Eq. 6.1 should be well-trained. This is especially important in ambiguous parts of the images such as those depicted by Fig. 5.2.

This necessitates significant amounts of ground-truth annotations to capture the large variability of the data and to cope with imaging artefacts and noise. To decrease the amounts of necessary time and effort, we introduce an AL algorithm that is suited to delineation problems represented on a graph. At each iteration it selects a sequence of consecutive edges from an overcomplete graph, such as the one described above, which should be labeled next in order to decrease the uncertainty in the most ambiguous image regions.

In theory the sequences could be of arbitrary length, that is 1,2, or more. In practice, we will see that 2 is near optimal because 2 consecutive edges are enough to capture some amount of geometry and because querying at each iteration more than 2 edges does not update the model frequently enough.

In the results section, we will use the algorithm of [110], which operates on edge pairs to



(a) Active Learning workflow (b) Performance change in AL

Figure 5.3 – **Active Learning.** (a) A typical scenario: a large pool of unlabelled samples is available and an AL algorithm indicates which ones should be labelled in the next iteration. After the annotator labels selected examples the classifier is retrained. (b) As more and more examples are added to the training set, the performance of our model on a separate test set increases and at some point it reaches a plateau when the training can be finished.

produce the final delineations. However, our approach is generic and could be used in conjunction with any delineation pipeline that represents the problem on a graph and requires supervised edge classification.

5.3 Approach

In this section, we first cast the traditional Uncertainty Sampling approach into our chosen delineation framework. We then introduce our approach to probability propagation designed to rapidly identify ambiguous image regions and prevent the so-called sampling bias that may lead the classifier to explore irrelevant parts of the feature space. Finally, we combine this with an approach to batch density-based learning that simplifies the interaction while guaranteeing that the sample batches are representative and diverse enough to achieve rapid convergence.

5.3.1 Random and Uncertainty Sampling

A typical AL scenario is presented in Fig. 5.3. It involves a big pool of unlabelled data and an oracle that can provide labels for the samples that the AL method finds to be the most informative for our problem. The classification model is retrained at every iteration and the new predictions are used to identify the next example that should be labelled. As the training progresses, the performance of our model increases.

The simplest strategy for picking samples to be annotated is to randomly choose them from a

pool of unlabeled ones in so called Random Sampling (**RS**). Uncertainty Sampling (**US**) is a simple and popular approach to more efficient learning by querying first the most uncertain samples according to a metric, such as Shannon entropy.

In our case, as discussed in Section 5.2, each edge e_i of the spatial graph $\mathcal{G}$ is assigned a feature vector $\mathbf{x}_i$ computed from the pixels surrounding the corresponding path. The feature vectors associated to each path are based on Histogram of Oriented Gradients specially designed for linear structures. They capture the contrast, orientation, and symmetry of the paths. Let

$$p_t(y_i = y|\mathbf{x}_i) \text{ for } y \in \{0,1\} \tag{5.2}$$

be the probabilities computed by classifier C_t after t AL iterations that e_i lies on the centreline of a true structure or not. Let also S_t be the set of N_t annotated samples $(\mathbf{x}_j, y_j)_{1\leq j\leq N_t}$ used to train C_t. Next, p_0 denotes the probabilities returned by the classifier using the small initial batch S_0 of annotated samples. When training is complete after T iterations, p_T is then used to compute the probabilities that appear in Eq. 5.1.

Given a classifier C_{t-1} trained using the training set S_{t-1}, AL iteration t involves choosing one or more unlabeled edges, asking the user to label them, adding them to the training set S_{t-1} to form S_t and, finally, training classifier C_t. In **RS**, this is done by randomly picking one or more $\mathbf{x}$ not already in S_{t-1}. In **US**, it is done by computing for each $\mathbf{x}$ the entropy:

$$\begin{aligned} H(\mathbf{x}) \quad = \quad & -\log(p_{t-1}(y=0|\mathbf{x}))\, p_{t-1}(y=0|\mathbf{x}) \\ & -\log(p_{t-1}(y=1|\mathbf{x}))\, p_{t-1}(y=1|\mathbf{x}) \end{aligned} \tag{5.3}$$

and selecting the vector(s) with the highest entropy. Since $H(\mathbf{x})$ is largest when the classifier returns a 0.5 value and minimum when it returns values close to zero or one, this assumes that those vectors whose probability of being a true path is computed to be 0.5 are the most uncertain and closest to the decision boundary. Therefore annotating them is likely to help refine the shape of that boundary.

This approach can be effective but it can also fall victim to *sampling bias*. This happens when the current classifier is so inaccurate that its decision boundary is far away from the real one and the learner ends up focusing on an irrelevant part of the feature space. Our approach is designed to avoid this trap.

5.3.2 Probability Propagation

The probability p_t returned by the path classifier takes into account the appearance of only a single path. By doing so, it neglects the information present in the wider neighborhood, provided by the other paths in the graph that share an endpoint with it. In particular, it ignores the fact that contiguous paths are more likely to share labels than non-contiguous ones.

To account for this, we took inspiration from the semi-supervised learning method of [123] and implemented a modified version of it that propagates probabilities instead of labels. There, the label propagation is used to classify a large pool of unlabelled examples having only a few labelled instances. In our Probability Propagation Sampling (**PPS**) strategy we propagate the probabilities assigned by the base classifier to identify samples that differ significantly from their neighbourhood.

Let $\mathbf{P}_0$ be an $N \times 2$ matrix. Its entries are the probabilities $p_t(y_i = y|\mathbf{x}_i)$ of Eq. 5.2 for all N samples and $y \in \{0,1\}$, except for already annotated ones for which we clamp the values to zero or one depending on their label. The information is then propagated as follows:

1. Build an $N \times N$ affinity matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ with elements $w_{ij} = \exp(-||\mathbf{x_i} - \mathbf{x_j}||^2/2\sigma^2)$ if e_i and e_j share a node and zero otherwise.
2. Build a symmetric matrix $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is diagonal with elements $d_{ii} = \sum_j w_{ij}$.
3. Iterate $\mathbf{P}^{i+1} = \alpha\mathbf{S}\mathbf{P}^i + (1-\alpha)\mathbf{P}_0$ followed by normalization of the rows of $\mathbf{P}^{i+1}$ until convergence, where $\alpha \in (0,1)$ specifies how much information is exchanged between neighbors and how much of the original information is retained. The series was shown to converge to $\mathbf{P}^* = (\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{P}_0$ [123] and we will use the closed-form solution.

After the probability propagation, we can compute the entropy of each path at AL iteration t, but this time using the new estimates of probability p^*:

$$\begin{aligned} H(\mathbf{x}) \quad = \quad & -\log(p^*_{t-1}(y=0|\mathbf{x}))\, p^*_{t-1}(y=0|\mathbf{x}) \\ & -\log(p^*_{t-1}(y=1|\mathbf{x}))\, p^*_{t-1}(y=1|\mathbf{x}) \end{aligned} \tag{5.4}$$

5.3.3 Density-based Batch Query

The scheme of Section 5.3.2 involves retraining the classifier each time the user has annotated a new sample, forcing them to wait for the computation to be over before intervening again.

As discussed in Section 2.3, this is impractical and most practical AL approaches work in batch-mode, that is, they allow the user to annotate several samples before retraining.

In our case, samples are image paths and it is much easier to sample several paths in the same image region than over a wide space, which would imply scrolling through a potentially large 2D image or, worse, 3D image stack. Our solution to this is to present the annotator with consecutive paths represented by adjacent edges in the spatial graph $\mathscr{G}$ of Section 5.2. However, in order to be effective, individual paths should be:

1. *informative* to ensure that the new labels truly bring new information,
2. *representative*, that is, inliers of the statistical distribution of all samples,
3. *diverse*, that is, different from each other and from the already labeled ones.

The entropy measure of Eq. 5.4 can be used to assess the first of these three desirable properties. To measure the other two, we use the $N \times N$ affinity matrix $\widetilde{\mathbf{W}}$ obtained using the same parameters as the matrix $\mathbf{W}$ of Section 5.3.2, but whose elements are measures of pairwise similarity between *each* of the N samples in the feature space, not only the neighbours in image.

Let L be the indices of already labelled edges and E_k be the set of all possible edge index combinations denoting k consecutive paths. For each set (batch) of consecutive edges $E \in E_k$, we can compute the following similarity measures:

$$\sigma_G(E) = \sum_{i \in E} \sum_{1 \le j \le N} w_{ij} \tag{5.5}$$

$$\sigma_L(E) = \sum_{i \in E} \sum_{l \in L} w_{il} \tag{5.6}$$

$$\sigma_I(E) = \sum_{i \in E} \sum_{j \in E, j \neq i} w_{ij}, \tag{5.7}$$

where $\sigma_G(E)$ is a global similarity measure, $\sigma_L(E)$ measures similarity to already labelled samples and $\sigma_I(E)$ similarity within the batch. Intuitively, we want to maximize σ_G to ensure representativeness and minimize σ_L and σ_I to improve diversity and explore the whole feature space. We therefore take

$$\mu(E) = \frac{\sigma_G(E) - \sigma_L(E) - \sigma_I(E)}{\sigma_G(E)}, \tag{5.8}$$

to be our non-negative measure of both diversity and representativeness. This formulation does not require any additional parameters to weigh the effect of the three conditions or constructing any additional graphs in the feature space.

5.3.4 Combining Informativeness and Density Measure

PPS allows us to take advantage of the current model while density-based query enables exploration of the feature space. In order to combine those two effects at each AL iteration we query the batch

$$E^* = \arg\max_{E \in E_k} -\mu(E)\left(\sum_{i \in E} H(\mathbf{x}_i)\right) \tag{5.9}$$

where H is the entropy measure of Eq. 5.4 and $\mu(E)$ is calculated as in Eq. 5.8. Combining **PPS** and the density measure μ leads to Density-Probability Propagation Sampling (**DPPS**), where the effects of exploration and exploitation are balanced during AL.

5.4 Results

In this section, we present our results; we first describe our experimental setup and baselines. We then introduce a synthetic dataset to help visualize the query decisions made by the different strategies. Finally, we show that our approach outperforms the conventional and state-of-the-art techniques on four real datasets.

5.4.1 Experimental Setup

We apply our AL approach for reconstruction of curvilinear networks in 2- and 3-D images. As discussed in Section 5.2, the overcomplete graphs, as well as the final delineations obtained once the classifiers have been properly trained are constructed using the delineation algorithm of [110].

The probabilities of Eq. 5.1 are computed by feeding the feature vectors to Gradient Boosted Decision Trees [9] with an exponential loss. We found it well suited to interactive applications because it can be retrained fast, that is in under 3s for all the examples we show in this chapter. To avoid overfitting especially in the initial stages of AL, we set the number of weak learners to 50, maximum tree depth to 2 and shrinkage to 0.06. Each tree is optimized using 50% of randomly selected data. Out of possible 303 features, 50 are investigated at each split. The classifier returns score F that can be then converted to probability using the logistic correction [79], that is

$$p(y = 1|x) = \frac{1}{1 + \exp(-2F(x))}\,. \tag{5.10}$$

The edge connectivity matrix of Section 5.3.2 is computed on the basis of the overcomplete graphs.

The annotated ground truth data we have for all datasets, allows us to simulate the user intervention. We assume edges that are 10 pixels/voxels apart from the corresponding ground-truth path and with a normalised intersection exceeding 0.5 to be positive. Otherwise they are treated as negatives. We start each query by a random selection of 4 data points belonging to each class (background/network). Unless stated otherwise, we query 2 consecutive paths during each iteration and this choice is explained in Section 5.4.4. We proceed until the total number of labelled samples reaches 100. Each AL trial is repeated 30 times and the results are then averaged.

5.4.2 Baselines

We compare the two versions of our approach, Probability Propagation Sampling (**PPS**) and Density Probability Propagation Sampling (**DPPS**) as described in Sections 5.3.2 and 5.3.4, to the following baselines:

- Random Sampling (**RS**) - selecting a random pair at each iteration.
- Uncertainty Sampling (**US**) - selecting a pair with the highest sum of individual entropies as given by Eq. 5.3.
- Query-By-Committee (**QBC**) - selecting a pair that causes the greatest disagreement in a set of hypotheses, here represented by trees in a Random Forest. We measure the disagreement using the definition of [18].

Moreover, we compare the real datasets also to the following state-of-the art methods:

- Information Density (**ID**) [95] - similarly to our method it combines uncertainty and density terms to select the next sample. Uncertainty is computed using probability entropy and density is approximated by computing average cosine similarity of every sample to the rest of unlabelled data.
- Reinforcement Active Learning Formulation (**RALF**) [24] - combines AL and reinforcement learning that allows for time-varying trade-off between exploration and exploitation. It adapts the sampling strategy during the learning process and can handle multiple selection criteria by framing the learning as MArkov Decision Process.

For calibration purposes, we also report the classification performance using all the available training data at once (**Full**), that is, without any AL.

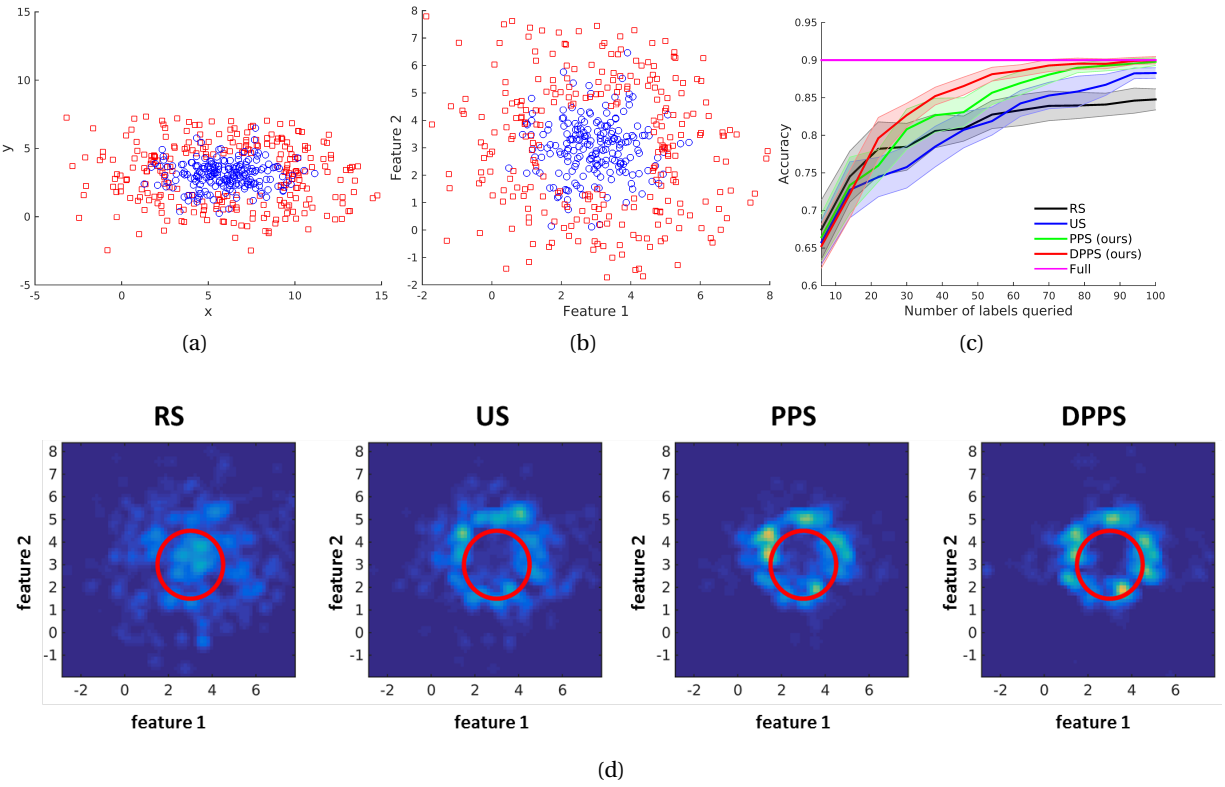


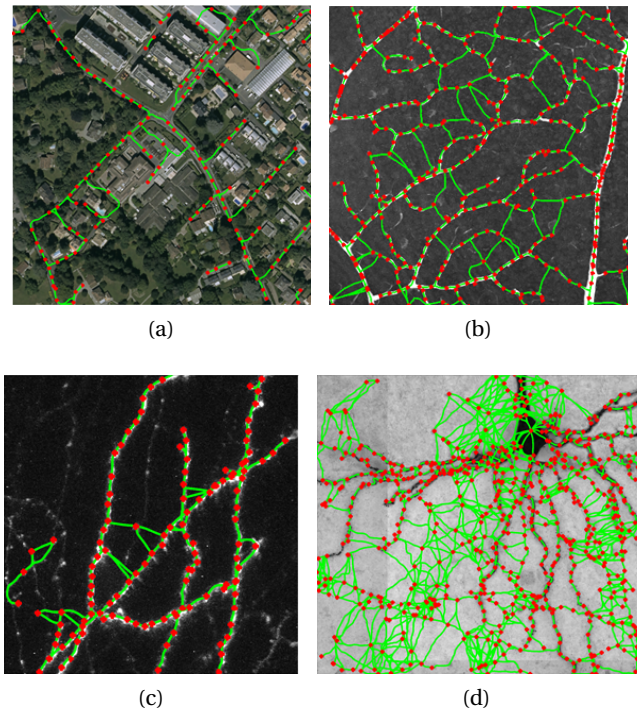
Figure 5.4 – **Synthetic dataset.** (a) Samples in the Euclidean space. (b) Samples in the feature space. (c) Classification results. (d) Query heat-maps in the feature space; the red circle indicates the optimal decision boundary. Best viewed in color.

5.4.3 Synthetic Dataset

To compare the qualitative behavior of different strategies, we create a synthetic dataset. In the image space depicted by Fig. 5.4a, a positive class is surrounded by a negative one, which resembles what happens when trying to find real linear paths surrounded by spurious ones. We created feature space depicted by Fig. 5.4b by transforming the image coordinates and adding random noise so that the decision boundary in feature space does not correspond to the one in Euclidean space. We built the required spatial graph by connecting each point to its 10 nearest-neighbors in image space. We compute the weighting matrix $\mathbf{W}$ using RBF kernel with $\sigma = 1$ and set probability propagation α to 0.9.

As can be seen in Fig. 5.4c, **PPS** and **DPPS** outperform the baselines and after querying 90 examples match the performance obtained by training on the whole training set. This corresponds to a 80% reduction in annotation effort. Furthermore, **DPPS** does better than **PPS** early on.

In Fig. 5.4d, we use a heat map in feature space to depict the the most frequently queried



(a) (b) (c) (d)

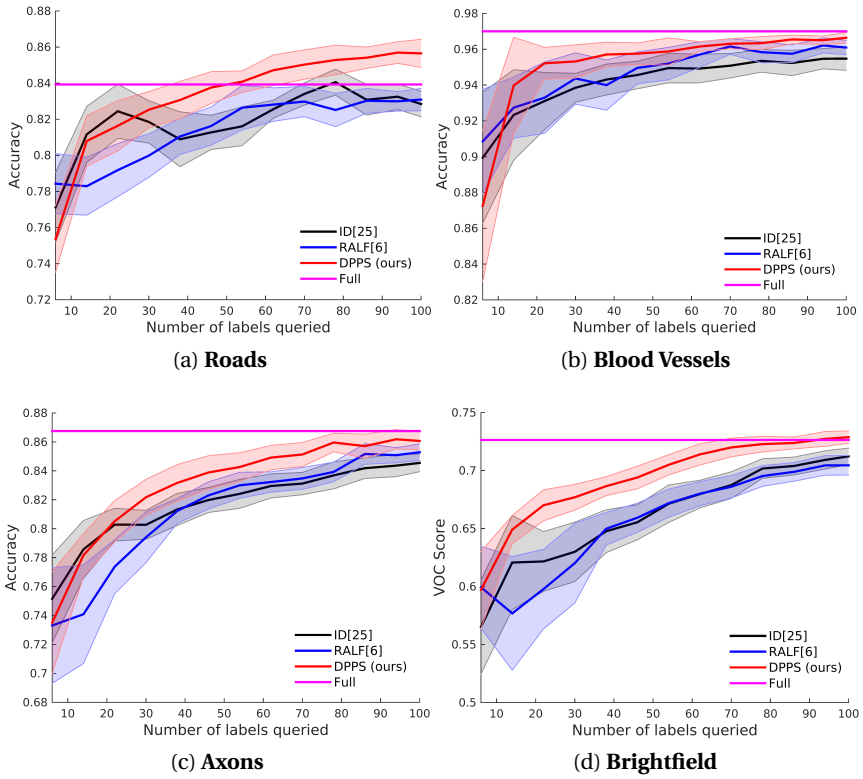
Figure 5.5 – **Datasets.** Training images with superimposed overcomplete graphs. (a) *Roads* (b) *Blood vessels* (c) *Axons* (d) *Brightfield neurons*.

regions and overlay the optimal decision boundary in red. They indicate that propagating information in a spatial graph helps refine the search space faster than simple uncertainty query. Introducing density measures further constrains the search space making the process more effective and sampling more uniformly around the optimal decision boundary.

5.4.4 Real Datasets

Roads The dataset consists of 2D aerial images of roads. They include road patches occluded by trees and contain road-like structures such as driveways, thus making the classification task difficult.

We compute the weighting matrix **W** using RBF kernel with $\sigma = 1$ and set probability propagation α to 0.9. The graph is constructed using only training data and during the whole AL process the classifier does not have access to test data. As shown in Fig. 5.6a and Table 5.1, both our approaches outperform the baselines and reach the full-dataset performance after as few as 50 samples, which corresponds to 75% reduction in annotation effort. Interestingly, the accuracy keeps increasing above the **Full** dataset accuracy. This behavior was already reported in [91] and suggests that in some cases a well chosen subset of data produces better



(a) **Roads** (b) **Blood Vessels**

(c) **Axons** (d) **Brightfield**

Figure 5.6 – **Quantitative results.** Path classification results after AL queries for the (a) *Roads* (b) *Blood vessels* (c) *Axons* (d) *Brightfield neurons* datasets. Shaded area corresponds to one standard deviation.

generalization than the complete set. As can be seen in Fig. 5.7 as the training progresses, the samples become more and more certain, decreasing the entropy. This is also when the probability propagation strategy proves especially useful compared to simple **US**. The analysis of the most frequently queried samples shown in Fig. 5.8a reveals that our method selects mainly the occluded paths and those at the intersections between two roads of different sizes or a road and a driveway. They correspond to the ambiguous cases discussed in Section 5.2 and presented in Fig. 5.2c and Fig. 5.2d. This makes it possible to learn the correct connectivity pattern and avoid mistakes as we postulated in Section 5.2. To verify this, we compare not only the classification performance, but also the quality of the final reconstruction. We run the full reconstruction framework with classification followed by an optimization step and evaluate the reconstruction using the DIADEM score [7]. It ranges from 0 to 1 with 1 being a perfect reconstruction. As shown in Fig. 5.9a, our approach outperforms the baselines also in terms of the quality of the final reconstruction. Interestingly, we again get a better result than by training with the **Full** dataset.

These results were obtained by querying pairs of edges. To test the influence of the length

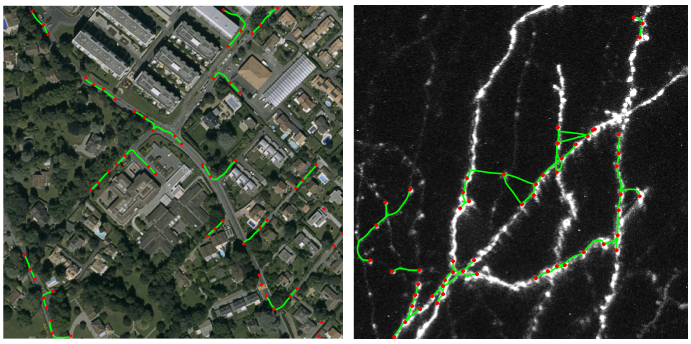
	RS	US	QBC	ID	RALF	PPS	DPPS
Roads	0.808	0.817	0.821	0.822	0.816	0.825	**0.835**
Blood vessels	0.942	0.944	0.955	0.943	0.948	0.953	**0.956**
BF neurons	0.625	0.653	0.646	0.665	0.658	0.673	**0.697**
Axons	0.818	0.821	0.811	0.821	0.816	0.830	**0.836**

Table 5.1 – Quantitative results. Area under the learning curve for all tested methods. An example of such learning curve is depicted in Fig. 5.6d

Figure 5.7 – **Learning progress.** Color-coded ambiguity of the training paths given by Eq. 5.9 after 5, 15, 25, 35 and 45 AL iterations. The measure was normalized across the iterations. There is a clear decrease in the uncertainty as the learning progresses.

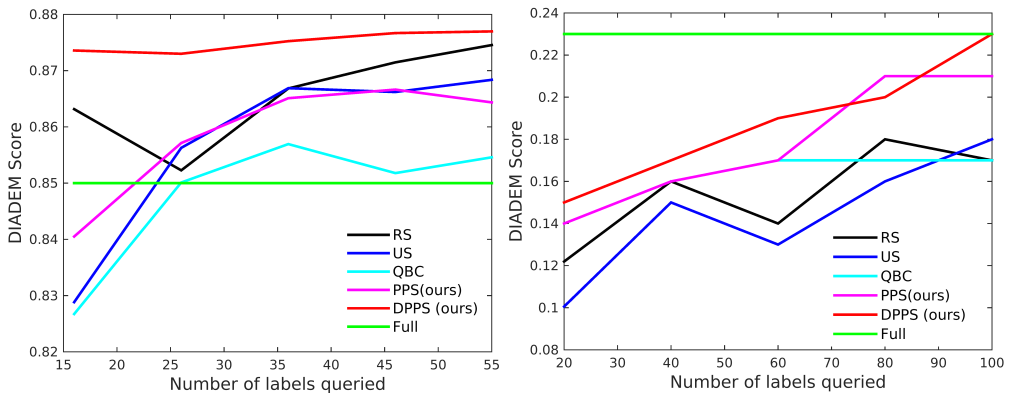
of the sequences we query, as discussed at the end of Section 5.2, we reran the experiments using singletons, pairs, and triplets. As can be seen in Fig. 5.10, using pairs tends to give the best results and this is what we will do in the remainder of this chapter. Note that we assume that annotating one edge counts as one label, but in reality the effort of annotating several *consecutive* edges is less than labeling the same number of instances at random locations, as the user does not need to scroll from one region to another.

Blood vessels The image stacks depicting direction-selective retinal ganglion cells were acquired with confocal microscopes. They contain many cycles and branch crossings. We compute the weighting matrix $\mathbf{W}$ using RBF kernel with $\sigma = 0.7$ and set α to 0.9. As shown in Fig. 5.6b, our two methods bring about improvements, especially at the beginning of AL.



(a) (b)

Figure 5.8 – **Visualization of frequent queries.**The most frequently queried samples for the (a) *Roads* and (b) *Axons* datasets. They often coincide with the ambiguous cases as discussed in Section 5.2.



(a) **Roads** (b) **Axons**

Figure 5.9 – **Effect of AL on final reconstruction.** Averaged DIADEM scores of final reconstruction for the (c) *Roads* and (d) *Axons* datasets.

Axons dataset consists of 3D 2-photon microscopy images of axons in a mouse brain. The main challenge associated with these images is low resolution in the z-dimension resulting in some disjoint branches being merged into one, which drastically changes the connectivity of the final solution.

We compute the weighting matrix **W** using RBF kernel with $\sigma = 3$ and set α to 0.9. The accuracy plot (Fig. 5.6c) reveals that yet again our method performs better than the baselines, especially in the later stages of learning, and result in a 65% reduction in the training effort. As seen in Fig 5.8b, the most frequently queried edges are concentrated in the regions where two branches seem to intersect in the xy-plane. In Fig. 5.9b we show that this again improves the quality of the final reconstruction.

	RS	US	QBC	ID	RALF	PPS	DPPS
Roads	0.023	0.020	0.023	0.019	0.019	0.019 6	**0.018**
Blood vessels	0.023	0.027	0.017	0.016	0.019	**0.014**	**0.014**
BF neurons	0.063	0.041	0.028	0.033	0.035	0.026	**0.017**
Axons	0.024	0.025	0.022	**0.018**	0.021	0.022	0.021

Table 5.2 – Standard deviation of the results. Generally our proposed methods have lower variance than baselines.

Brightfield neurons The dataset consists of 3D images of neurons from biocytin-stained rat brains acquired using brightfield microscopy. As in the *Axons* dataset, the z-resolution is low. The corresponding training graph is much bigger than in the previous 2 cases and consists of more than 3000 edges, most of which are negative. To assess the performance of different methods, we compute the VOC score [26] instead of accuracy. This is due to the fact that in this dataset around 95% of the edges are negative and the VOC score does not take into account true negatives. We compute the weighting matrix **W** using RBF kernel with $\sigma = 1$ and set α to 0.9. As seen in Fig. 5.6d and Table 5.1, our methods outperform the baselines. For **RALF**, we can notice the possible effects of bias trap, when the performance does not change for a few iterations, even though more and more labels are queried.

Note that each of the experiments was repeated 30 times and the results are averaged. In Table 5.2 we present also the variance of the results. In all but one cases except for one **PPS** approach shows smaller variance than the baselines and **DPPS** yields even lower variance.

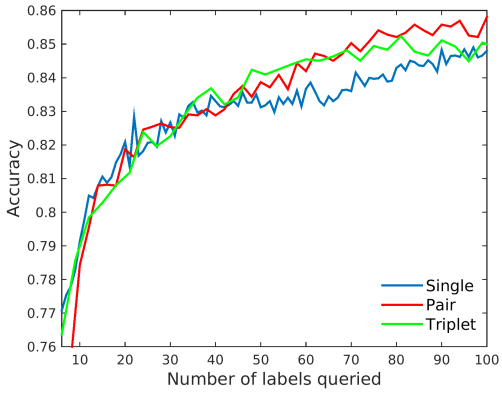


Figure 5.10 – **Effect of batch size.** The classification performance for different batch sizes for the *Roads* dataset.

5.5 Conclusion

In this chapter we introduced an approach to incorporating the geometrical information that increases the effectiveness of AL for the delineation of curvilinear networks. Additionally, we introduced a density-based strategy, which ensures that the selected batches are informative, diverse and representative of the underlying distribution. It also allows us to query sequences of consecutive paths, further reducing the annotation effort. Our approach showed superior performance for a wide range of networks and imaging modalities when compared to a number of conventional methods.

6 Active Learning and Proofreading Using Global Delineation Constraints

As we described in the previous chapter, many of delineation algorithms require manually annotated training data, which is tedious to obtain. Furthermore, even minor classification errors may significantly affect the topology of the final result and therefore the automatic reconstruction is often followed by manual proofreading. In this chapter we propose a generic approach to addressing both of these problems by taking into account the influence of a potential misclassification on the resulting delineation. In an Active Learning context, we identify parts of linear structures that should be annotated first in order to train a classifier effectively. In a proofreading context, we similarly find regions of the resulting reconstruction that should be verified in priority to obtain a nearly-perfect result. In both cases, by focusing the attention of the human expert on potential classification mistakes which are the most critical parts of the delineation, we reduce the amount of required supervision. We demonstrate the effectiveness of our approach on microscopy images depicting blood vessels and neurons.

Part of the material presented in this Chapter appeared in [75].

6.1 Introduction

As described in the previous chapter, many state-of-the-art approaches to automatically delineating com[plex curvilinear structures rely on supervised Machine Learning techniques. For training purposes, they require *annotated* ground-truth data in large quantities to cover a wide range of potential variations due to imaging artifacts and changes in acquisition protocols. For optimal performance, these variations must be featured in the training data, as they can produce drastic changes in appearance. Furthermore, no matter how well-trained the algorithms are, they will continue to make mistakes, which must be caught by the user and corrected. This is known as *proofreading* – a slow, tedious and expensive process when large amounts of image data or 3D image stacks are involved, to the point that it is considered

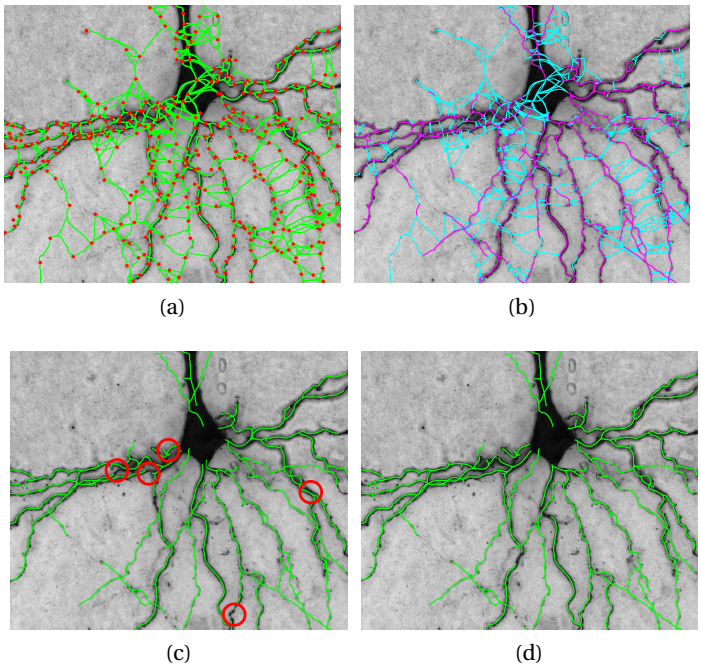
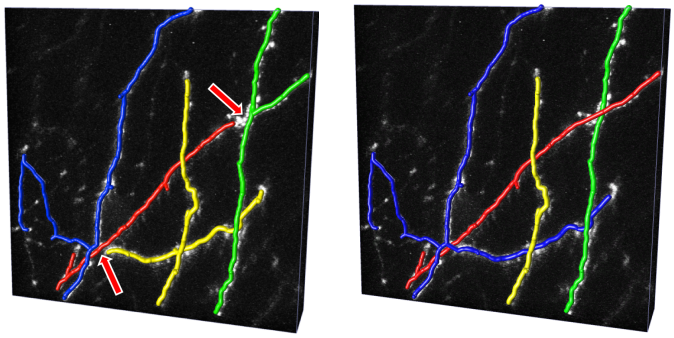


Figure 6.1 – **Delineation workflow.** (a) Input image with overcomplete graph overlaid. (b) The high-probability edges are shown in purple and the others in cyan. (c) Automated delineation, with connectivity errors highlighted by red circles. (d) Final result after proofreading.

as a major bottleneck for applications such as neuron reconstruction [85].

In other words, human intervention is required both to create training data before running the delineation algorithm and to correct its output thereafter. Current approaches to making this less tedious focus on providing better visualization and editing tools [22, 85]. While undoubtedly useful, this is not enough. We therefore propose an Active Learning (AL) [92] approach to direct the annotator's attention to the most critical samples. It takes into account the expected change in reconstruction that can result from labeling specific paths. It can be used both for fast annotation purposes and, later, to detect potential mistakes in machine-generated delineations.

More specifically, consider an algorithm such as those of [90, 120, 109, 77, 84], whose workflow is depicted by Fig. 6.1. It first builds a graph whose nodes are points likely to lie on the linear structures and whose edges represent paths connecting them. Then it assigns a weight to each edge based on the output of a discriminative classifier. Since the result is critically dependent on the weights, it is important that the classifier is trained well. Finally, the reconstruction algorithm finds a subgraph that maximizes an objective (cost) function dependent on the edge weights, subject to certain constraints. However, even very small mistakes can result in very different delineations, as shown in Fig. 6.2.



(a) (b)

Figure 6.2 – **Effect of small mistakes.** Misclassifying even a few edges may severely impact the final topology. (a) The two edges indicated by the red arrows are falsely labeled as negatives. As a result, two pairs of unrelated branches (green and yellow) are merged. (b) The true connectivity is recovered after correcting the two edges.

Our main insight is that the decision about which edges to annotate or proofread should be based on their influence on the cost of the network. Earlier methods either ignore the network topology altogether [28] or only take it into consideration locally [74], whereas we consider it globally. Our contribution is therefore a cost- and topology-based criterion for detecting attention-worthy edges. We demonstrate that this can be used for both AL and proofreading, allowing us to drastically reduce the required amount of human intervention when used in conjunction with the algorithm of [109]. To make it practical for interactive applications, we also reformulate the latter to speed it up considerably – it runs nearly in real-time and it can handle much larger graphs than [109].

The remainder of this chapter is organized as follows: first, in Section 6.2, we describe our attention mechanism for selecting important edges in the delineation. In Section 6.3 we explain how our attention mechanism a can be used for Active Learning and proofreading purposes. Then, in Section 6.4, we introduce a new, more efficient formulation of the state-of-the-art Mixed Integer Programming delineation algorithm that ensures fast and reliable reconstruction, which is necessary for online applications. Finally, in Section 6.5, we compare the performance of our algorithm against conventional techniques.

6.2 Attention Mechanism

6.2.1 Graph-Based Delineation

Delineation algorithms usually start by computing a tubularity measure [55, 108, 102], which quantifies the likelihood that a tubular structure is present at a given image location. Next, they extract either high-tubularity superpixels likely to be tubular structure fragments [90, 120] or

longer paths connecting points likely to be on the centerline of such structures [35, 11, 77, 109]. Each superpixel or path is treated as an edge e_i of an over-complete spatial graph $\mathscr{G}$ (see Fig. 6.1(a)) and is characterized by an image-based feature vector $\mathbf{x}_i$. Let $\mathscr{E}$ be the set of all such edges, which is expected to be a superset of the set $\mathscr{R}$ of edges defining the true curvilinear structure, as shown in Fig. 6.1(d). If the events of each edge e_i being present in the reconstruction are assumed to be independent (conditional on the image evidence $\mathbf{x}_i$), then the most likely subset $\mathscr{R}^*$ is the one minimizing

$$c(\mathscr{R}) = \sum_{e_i \in \mathscr{R}} w_i, \text{ with } w_i = -\log \frac{p(y_i = 1|\mathbf{x}_i)}{p(y_i = 0|\mathbf{x}_i)}, \tag{6.1}$$

where $w_i \in \mathscr{R}$ is the *weight* assigned to edge e_i and y_i is a binary class label denoting whether e_i belongs to the final reconstruction or not. This optimization is subject to certain geometric constraints; for example, a state-of-the-art method presented in [109] solves a more complex Mixed Integer Program (**MIP**), which uses linear constraints to force the reconstruction to form a connected network (or a tree). As described in Section 6.4, we were able to reformulate the original optimization scheme and obtain major speedups which make it practical even when delineations must be recomputed often. In Section 6.5 we also show that it yields better results than using a more basic method Minimum Spanning Tree with Pruning [35], while also being able to handle non-tree networks. Let us remark that finding the minimizing $\mathscr{R}$ is trivial to parallelize.

The probabilities appearing in Eq. 6.1 can be estimated in many ways. A simple and effective one is to train a discriminative classifier for this purpose [11, 120, 109]. However, the performance critically depends on how well-trained the classifier is. A few misclassified edges can produce drastic topology changes, affecting the whole reconstruction, as shown in Fig. 6.2. In this chapter we address both issues with a single generic criterion.

6.2.2 Error Detection

The key to both fast proofreading and efficient AL is to quickly find potential mistakes, especially those that are critical for the topology. In this work, we take *critical mistakes* to mean erroneous edge weights w_i that result in major decrease in the cost $c(\mathscr{R}^*, \mathbf{W})$ of the reconstruction. In other words, if changing a specific weight can significantly influence the delineation, we must ensure that the weight is correct. We therefore measure this influence, alter the edge weights accordingly, and recompute the delineation.

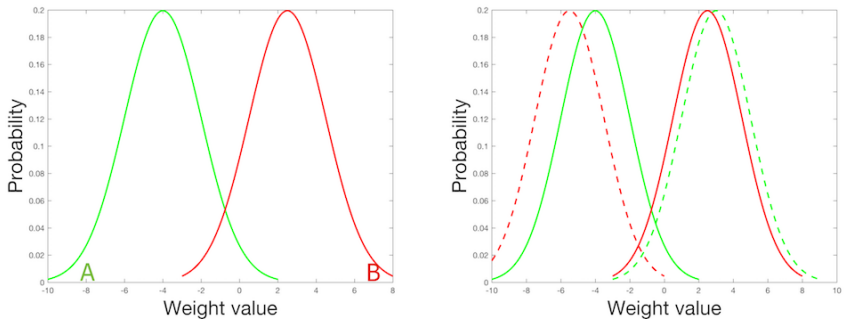


Figure 6.3 – **Changing the weights.** (a) Two Gaussian distributions corresponding to positive (green) and negative (red) classes of edges. (b) The effect of weight transformation; the original distributions are drawn with solid lines, while the corresponding distributions after the transformation are drawn with dashed lines. The described transformation causes "swapping" of the distributions corresponding to the two classes.

Delineation-Change Metric

We denote by $\mathscr{R}^*$ the edge subset minimizing the objective (cost) function $c(\mathscr{R}, \mathbf{W}) = \sum_{e_i \in \mathscr{R}} w_i$ given a particular set $\mathbf{W}$ of weights assigned to edges in $\mathscr{G}$. Changing the weight w_i of edge e_i to w_i' will lead to a new graph with optimal edge subset $\mathscr{R}_i'$. We can thus define a delineation-change metric, which evaluates the cost of changing the weight of an edge $e_i \in \mathscr{E}$:

$$\Delta c_i = c(\mathscr{R}^*, \mathbf{W}) - c(\mathscr{R}_i', \mathbf{W}') \,. \tag{6.2}$$

If $\Delta c_i > 0$, the cost has decreased; we can conjecture that the overall reconstruction benefits from this weight change and therefore the weight value may be worth investigating by the annotator as a potential mistake. The converse is true if $\Delta c_i < 0$. In other words, this very simple metric gives us a way to gauge the influence of an edge weight on the overall reconstruction.

Changing the Weights

For our cost change criterion to have practical value, we must alter weights in such a way that Δc_i is largest for edges which require the opinion of an annotator. In practice, the weights of positive-class edges tend to follow a Gaussian distribution with negative mean and a variance such that few of them are positive values, as shown in Fig. 6.3(a). Similarly, negative edges follow a Gaussian distribution with positive mean, few of them being negative. As a result, most of the mistaken edges have $|w_i| \approx 0$.

In order for our delineation-change metric to be informative, we must ensure that attention-

worthy edges (probable mistakes) have high values of Δc_i. To achieve this, we must not only flip the sign of the weight (implying assigning it to the opposite class), but also increase the absolute value of likely mistakes weights. Without this, many of the mistakes with $|w_i| \approx 0$ could be omitted due to smaller values of Δc_i compared to edges with weights of higher absolute value, which are much less likely to be mistakes.

The above requirements can be satisfied with the following transformation:

$$w_i' = \begin{cases} A + w_i & \text{if } w_i > 0, \\ B + w_i & \text{if } w_i < 0. \end{cases} \tag{6.3}$$

It is equivalent to swapping the distributions corresponding to positive and negative edges, as shown in Fig 6.3(b).

We take A and B to be the 10% and 90% quantiles of the weight distribution (for robustness to outliers). These are near-extreme values of the weights for the positive and negative classes respectively, which we use as attractors for w_i': for small positive w_i we want w_i' to be close to A, and for negative ones to B instead. The weight change is therefore likely to yield a significant Δc_i for probable mistakes.

Finally, for edges whose weight is negative but which nevertheless do not belong to the graph, we take Δc_i to be w_i' to ensure that it is positive and that more uncertain edges are assigned higher Δc_i.

6.3 Active Learning and Proofreading

AL aims to train a model with minimal user input by selecting small subsets of examples that are the most informative. Formally, our algorithm starts with a small set of labeled edges S_0. We then repeat the following steps: At iteration t, we use the annotated set of edges S_{t-1} to train classifier C_{t-1} and select one or more edges to be labeled by the user and added to S_{t-1} to form S_t. The edge(s) we select are those that maximize the criterion Δc of Eq. 6.2.

By contrast, proofreading occurs *after* the classifier has been trained and a complete delineation has been produced. At this point, the main concern is not to further improve the classifier, but simply to correct potential mistakes. Therefore, the most crucial edges are those that are misclassified and whose presence or absence most affects the topology of the delineation. To find them, we again compute the Δc value for each edge. However, some edges could have a high Δc because they are misclassified, even though they do not influence the topology of the final delineation.

To focus on potential mistakes that do affect the topology strongly, we rely on the DIADEM score [7], which captures the topological differences between trees, such as connectivity changes and missing or spurious branches. It ranges from 0 to 1; the larger the score, the more similar the two trees are. In Appendix we present details about DIADEM computation.

In this case, let $\mathscr{R}^*$ be the optimal tree given the edge weights, and let $\mathscr{R}'_i$ be the tree we obtain when changing the weight of edge e_i from w_i to w'_i, as described in Section 6.2.2. To measure the importance of each edge, we compute the score

$$s_i = \frac{\Delta c_i}{\text{DIADEM}(\mathscr{R}^*, \mathscr{R}'_i)} \tag{6.4}$$

and ask the user to check the highest-scoring one. The edge is assigned a weight equal to A or B from Section 6.2.2 according to the user's response. We then recompute $\mathscr{R}^*$ and repeat the process. Note that this is very different from traditional proofreading approaches, which require the user to visually inspect the whole image. By contrast, our user only has to give an opinion about one edge at a time, which is automatically selected and presented to them.

6.4 Fast Reconstruction of Curvilinear Structures

AL and proofreading parts of our pipeline have to operate in online settings in order to be truly useful. At the same time our attention approach requires re-computing optimal reconstructions many times. However, the previous formulation of graph-based delineation in [109] may take even up to few hours to converge. In this section we describe reformulation of this problems, which significantly decreases its size and operates in real-time, making it possible to employ in interactive mode.

Reconstruction of networks of curvilinear structures involves solving the following problem:

Min-Weight Tree Containing r (MinTree)

Given: A graph $\mathscr{G} = (V, \mathscr{E})$, a root vertex $r \in V$, weights on edges $w : \mathscr{E} \to \mathbb{R}$. Weights may be negative.

Find: A tree $\mathscr{R} \subseteq \mathscr{G}$ containing the vertex r, minimizing the sum of weights of picked edges $\sum_{e \in \mathscr{R}} w(e)$.

In our approach, MinTree is used when we expect the ground-truth image to be a tree. If such

an assumption is not realistic (loopy networks, such as blood vessels), then we are instead interested in the following problem MinSubgraph:

Min-Weight Connected Subgraph Containing r (MinSubgraph)

Given: A graph $\mathscr{G} = (V, \mathscr{E})$, a root vertex $r \in V$, weights on edges $w : \mathscr{E} \to \mathbb{R}$. Weights may be negative.

Find: A connected subgraph $\mathscr{R} \subseteq \mathscr{G}$ which contains the vertex r (and is not necessarily a tree), minimizing the sum of weights of picked edges $\sum_{e \in \mathscr{R}} w(e)$.

Both problems are significantly harder than the Minimum Spanning Tree problem, because $\mathscr{R}$ does not need to connect the entire graph and also the weights may be negative. In fact, both problems are NP-complete;

In both [109] and our approach they are solved using a Mixed Integer Programming (**MIP**) formulation, which is given as input to the Gurobi solver.[1]

However, the previously considered formulation (see the model Arbor-IP in [109] and also the model M-DG in [10]) has $|V||\mathscr{E}|$ variables and as many constraints. This makes solving it costly for small graphs and impossible for larger ones. Our contribution is a new, linear-size **MIP** model for this problem, described below.

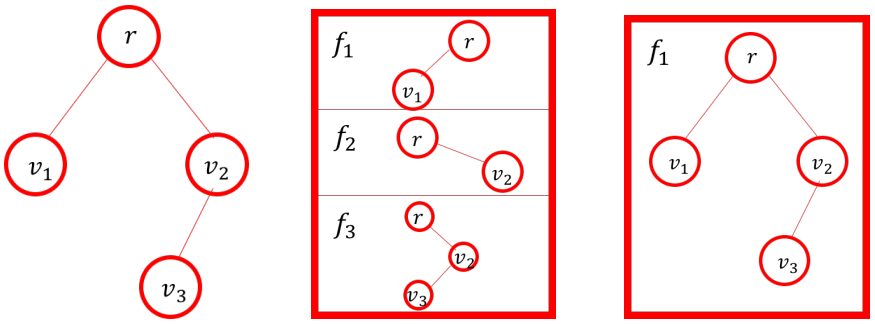
6.4.1 Our Formulation

First, we describe how to obtain a **MIP** for MinTree. We replace each undirected edge with two directed edges, so as to work with a directed graph. Our objective is to find a directed tree whose each edge is directed away from the root r (a so-called r-arborescence).

We associate a binary variable $x_{uv} \in \{0,1\}$ with each directed edge $(u,v) \in \mathscr{E}$, denoting the presence of the edge in the solution $\mathscr{R}$. The first two linear constraints to consider are:

- any vertex v has at most one incoming edge (r has none) (see equations (6.5–6.6) below),
- an edge (u,v) can be in the solution only if u has an incoming edge in the solution (or $u = r$) (6.7).

[1] [109] also introduce a more advanced algorithm, which uses a formulation with quadratic weights, i.e., weights on pairs of adjacent edges, rather than a linear weight function; this makes the computational burden even heavier.



Graph (a) | Flow formulation of [109] (b) | Our flow formulation (c)

Figure 6.4 – **Efficient MIP formulation.** Given an example graph (a) the formulation of [109] creates a separate flow variable from root r to any vertex v (b). We replaced it with a single flow from root to a set of sinks (c) thus reducing the size of **MIP** formulation.

These conditions almost require the solution to be an r-arborescence, but not quite; namely, there can still appear directed cycles (possibly with some adjoined trees). One way to deal with this issue is to enforce that every non-isolated vertex is connected to the root; this can be done using network flows. The constraints in the previous formulation require that, for every v with an incoming edge, there should exist a flow $\{f_e^v\}_{e\in\mathscr{E}}$ of value 1 from r to v, as shown in Fig. 6.4(b). However, this leads to a large program ($|V||\mathscr{E}|$ variables).

Our way around this is to instead require the existence of a single flow $\{f_e\}_{e\in\mathscr{E}}$ from the source vertex r to some set of sinks (Fig. 6.4(c)). The main constraints are that:

- for every vertex $v \neq r$, if v has an incoming edge (i.e., v is not an isolated vertex in the solution, but is spanned by $\mathscr{R}$), then the inflow into v is at least 1 more than the outflow (otherwise it is greater or equal to the outflow) (6.8),
- f is supported only on the support of x (that is, the flow f only uses edges which are used by the solution $\mathscr{R}$) (6.9).

Since x has no edges into the root, neither does f. Thus f is indeed a flow (within the x-subgraph) from the source r to the sink set being the set of all active vertices.

We write down our **MIP** formulation below. We use the following notation: $x(F) = \sum_{e\in F} x(e)$ for a subset $F \subseteq \mathscr{E}$, $\delta^+(v)$ is the set of (directed) edges outgoing from vertex v, and $\delta^-(v)$ is the set of (directed) edges incoming into vertex v. Thus e.g. $f(\delta^+(v))$ is the total f-flow outgoing from vertex v.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(u,v)\in\mathscr{E}} w(u,v)x_{uv} & & \\
\text{subject to} \quad & x_{uv} \in \{0,1\} & \forall (u,v) \in \mathscr{E} & \\
& x(\delta^-(v)) \leq 1 & \forall v \in V \setminus \{r\} & (6.5)\\
& x(\delta^-(r)) = 0 & & (6.6)\\
& x_{uv} \leq x(\delta^-(u)) & \forall (u,v) \in \mathscr{E}, u \neq r & (6.7)\\
& f(\delta^-(v)) - f(\delta^+(v)) \geq x(\delta^-(v)) & \forall v \in V \setminus \{r\} & (6.8)\\
& f_{uv} \geq 0 & \forall (u,v) \in \mathscr{E} & \\
& f_{uv} \leq (|V|-1) \cdot x_{uv} & \forall (u,v) \in \mathscr{E}. & (6.9)
\end{aligned}
$$

6.5 Results

In this section we describe the experimental protocol used to evaluate our attention mechanism and we present the evaluation results of different stages as well as the complete pipeline.

6.5.1 Datasets

We tested our approach on 3-D image stacks depicting various structures shown in Fig. 6.5:

- **Blood Vessels**: two stacks of retinal blood vessels obtained with confocal microscope. One was used to training and the other one for testing.
- **Brightfield Neurons**: three image stacks acquired using brightfield microscopy from biocytin-stained rat brains.
- **Axons**: 2-photon laser scanning microscopy images of mouse neocortex neurons labeled with fluorescence used to analyze the learning progress.
- **Olfactory Projection Fibers**: confocal microscopy image stacks of the Drosophile fly taken from the DIADEM competition [7].

We rely on the algorithm of [109] for the initial overcomplete graphs and the corresponding edge features. To classify edges as being likely to be part of an extended linear structure or not on the basis of local image evidence, we use Gradient Boosted Decision Trees [9]. We use our faster fomulation of **MIP** to obtain the final delineations.

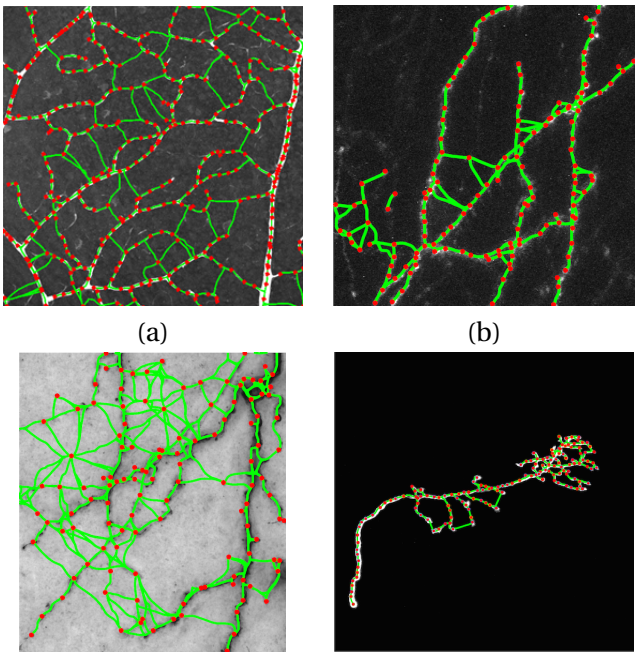


Figure 6.5 – **Datasets** Images used for evaluation with the over-complete graphs overlaid. (a) *Blood Vessels.* (b) *Axons.* (c) *Brightfield Neurons.* (d) *Olfactory Projection Fibers.*

6.5.2 Fast Reconstruction

We first empirically test our new **MIP** formulation described in Section 6.4. The optimization runtimes obtained using our formulation compared to the one presented in [109] are shown in Table 6.1 and Table 6.2. The optimization was executed on a 2x Intel E5-2680 v2 system (20 cores). Our formulation can be solved under 6 seconds for all real-world graph examples we have tried; the maximum for the formulation of [109] is over 6 minutes.

	Axon1	**Axon2**	**Axon3**	**Axon4**	**Axon5**	**Axon6**
# edges	164	223	224	265	932	2638
MIP [109]	0.91	1.04	1.19	1.45	78.3	393.7
MIP ours	0.03	0.10	0.04	0.23	0.10	5.23
speed-up	26.1x	10.1x	27.3x	6.3x	743.5	75.2x

Table 6.1 – Per-reconstruction runtimes (in seconds) of the **MIP** formulation of [109] and ours for the proofreading task on *Axons* dataset.

We also compared the runtimes on randomly generated graphs of various sizes – see Table 6.3. The speed-ups remain similar. In Table 6.4 we collect runtimes of our method on larger randomly generated graphs. If we assumed (more or less arbitrarily) 2 seconds to be the threshold of what is practical in an interactive setting (given that this optimization needs to be run multiple times), then we can see that the method of [109] can deal with graphs of size at

	BFNeuron1	BFNeuron2	OPF1	OPF2	BFNeuron3	BFNeuron4
# edges	120	338	363	380	645	2826
MIP [109]	0.48	2.25	1.53	1.65	2.13	308.23
MIP ours	0.02	0.12	0.05	0.08	0.26	2.30
speed-up	18.2x	17.7x	29.4x	19.9x	8.1x	134.0x

Table 6.2 – Per-reconstruction runtimes (in seconds) of the **MIP** formulation of [109] and ours for the proofreading task on *Brightfield Neurons* dataset.

most 300, whereas our method copes with graphs having around 2000 edges.

# edges	99	132	220	330	440	660	924	1320	1540
MIP [109]	0.16	0.30	1.13	3.39	8.35	29.35	73.16	112.59	149.01
MIP ours	0.03	0.04	0.06	0.12	0.15	0.29	0.36	0.67	0.42
speed-up	6.1x	7.5x	17.9x	29.4x	53.9x	102.8x	201.8x	167.8x	348.1x

Table 6.3 – Per-reconstruction runtimes (in seconds) of the **MIP** formulation of [109] and ours on random graphs.

# edges	1760	2420	3520	4400	5720	9900
MIP ours	1.60	2.71	6.59	9.57	15.52	81.55

Table 6.4 – Per-reconstruction runtimes (in seconds) of our **MIP** formulation on random graphs.

One further practical method for speeding up the solver is to initialize it with a nonzero feasible solution. In cases where we needed to explore a large number of reconstructions resulting from altering just one weight at a time (which was the setting of our method), we initialized the new solution to the current optimal solution. Note that this scenario makes performance considerations especially relevant, as $|\mathscr{E}|$ reconstructions need to be made; even though they can be run in parallel, a high running time of a single **MIP** solution would make the approach impractical.

6.5.3 Active Learning

Next, we tested our focusing mechanism described in Section 6.2 in Active Learning seetings. For each image, we start with an overcomplete graph. The initial classifier is trained using 10 randomly sampled examples. Then, we query four edges at a time, as discussed in Section 6.3, which allows us to update the classifier often enough while decreasing the computational cost. We report results averaged over 30 trials in Fig. 6.6. Our approach outperforms both naive methods such as Uncertainty Sampling (**US**) and more sophisticated recent ones such as **EMOC** [28] and our own approach presented in Chapter 5, which we denote **DPPS**. Compared to the method presented in this chapter, **DPPS** relies on uncertainty sampling, but only takes

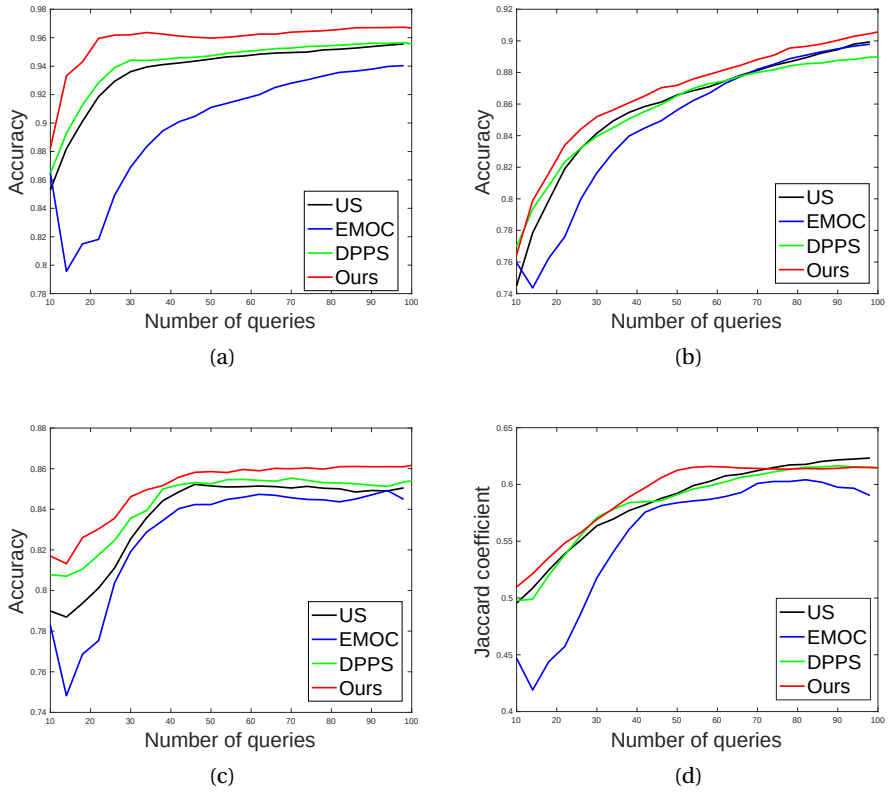


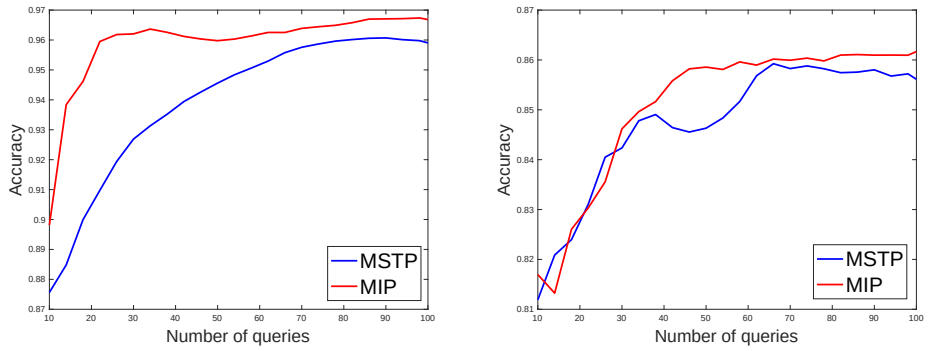
Figure 6.6 – **Active Learning.** Accuracy as a function of the number of annotated samples. (a) *Blood vessels.* (b) *Axons.* (c) *Brightfield neurons.* (d) *Olfactory Projection Fibers.* The red curve denoting our approach is always above the others, except in the right-hand side of (d): because this is a comparatively easy case, the delineation stops changing after some time and error-based queries are no longer informative.

local topology into account when evaluating this uncertainty. **EMOC** is a more generic method that aims at selecting samples that have the greatest potential to change the output.

In Fig. 6.7 we can see that using the **MIP** formulation indeed helps improve the AL results, compared to a more basic but equally fast method Minimum Spanning Tree with Pruning [35] (**MSTP**), as it produces more accurate reconstructions and thus we can more reliably detect mistakes. This is visible especially in case of **Blood Vessels**, which in reality can form loops. Those can be reconstructed using MinSubgraph **MIP**, but not with **MSTP**.

6.5.4 Proofreading

In order to evaluate attention approach in proofreading setting, we compute an overcomplete graph for each test image and classify its edges using a classifier trained on 20000 samples. We then find four edges with the highest values of the score s_i of Eq. 6.4 and present them to the



(a) (b)

Figure 6.7 – **MSTP vs. MIP.** Comparison of our AL strategy when using **MSTP** and **MIP** on (a) *Blood Vessels* (b) and *Brightfield Neurons.* datasets. Using **MIP** yields better results because it provides more precise reconstruction that is not constraint only to tree-like structures.

user for verification. Their feedback is then used to update the delineation.

The red curves of Fig. 6.8(a-c) depict the increase in DIADEM score. Rapid improvement can be seen after as few as 15 corrections. Fig. 6.9 shows how the reconstruction evolves in a specific case. For analysis purposes, we also reran the experiment using the Δc criterion of Eq. 6.2 (cost-only) instead of the more sophisticated one of Eq. 6.4 (cost and topology) to choose the paths to be examined. The green curves in Fig. 6.8(a-c) depict the results. They are not as good, particularly in the case of Fig. 6.8(c), because the highest-scoring mistakes are often the ones that tend to be in the **MIP** reconstruction both before and after correcting mistakes. It is therefore only by combining both cost and topology that we increase the chances that a potential correction of the selected edge will improve the reconstruction. By contrast, paths chosen by **RS** and **US** are not necessarily erroneous or in the immediate neighborhood of the tree. As a result, investigating them often does not give any improvements.

6.5.5 Complete Pipeline

In a working system, we would integrate AL and proofreading into a single pipeline. To gauge its potential efficiency, we selected 50 edges to train our classifier using the AL strategy of Section 6.3. We then computed a delineation in a test image and proofread it by selecting 35 edges. For comparison purposes, we used either our approach as described in Section 6.3, **RS**, or **US** to pick the edges for training and then for verification. In Fig. 6.8(d) we plot the performance (in terms of the DIADEM score of the final delineation and of the ground truth) as a function of the total number of edges the user needed to label manually.

In total, we estimated that combining three parts of our system (using Active Learning to

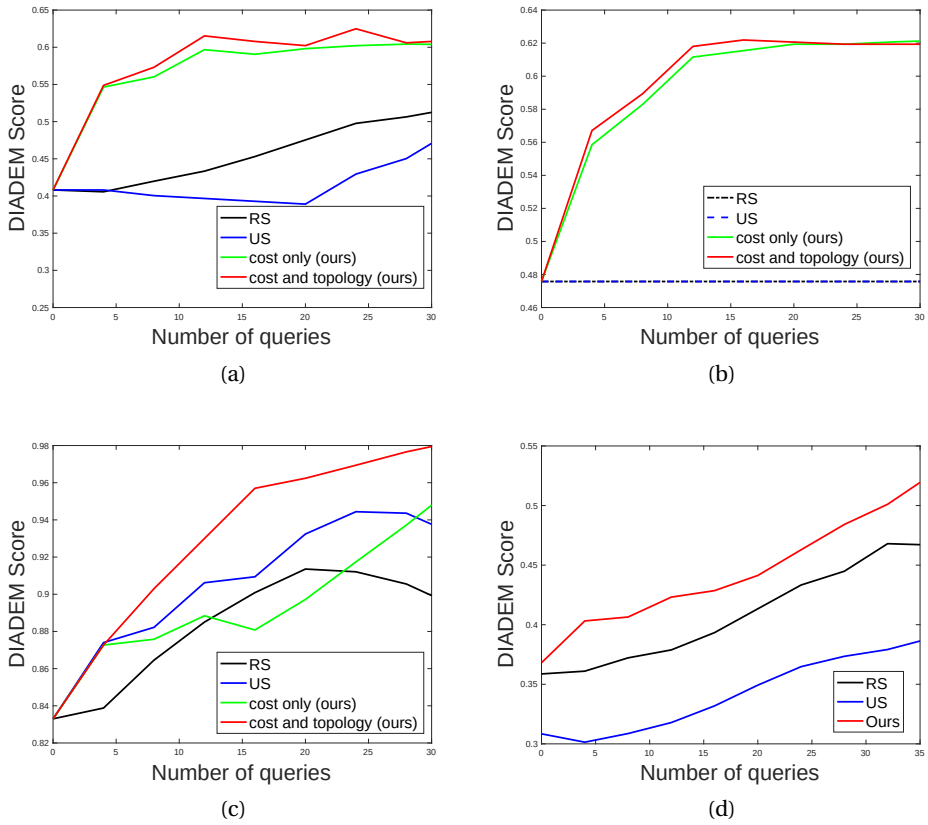


Figure 6.8 – **Focused proofreading.** DIADEM score as a function of the number of paths examined by the annotator. (a) *Axons*. (b) *Brightfield Neuron*. (c) *Olfactory Projection Fibers*. (d) Combined AL and proofreading for *Axons*.

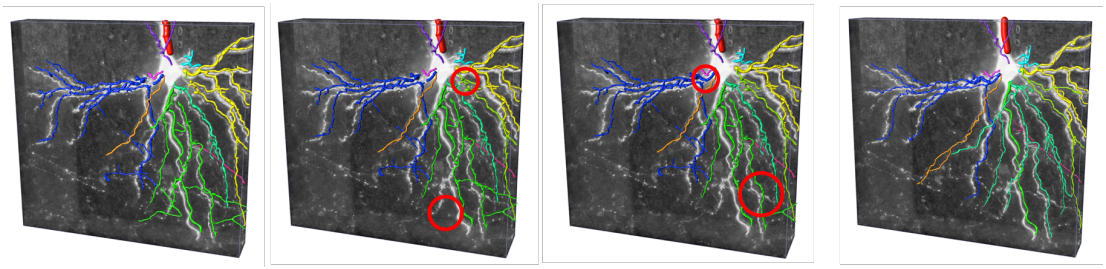


Figure 6.9 – **Proofreading.** From left to right: initial delineation, delineations after 10 and 20 corrections, and ground truth. Note changes in connectivity (in red circles) as more and more samples are corrected

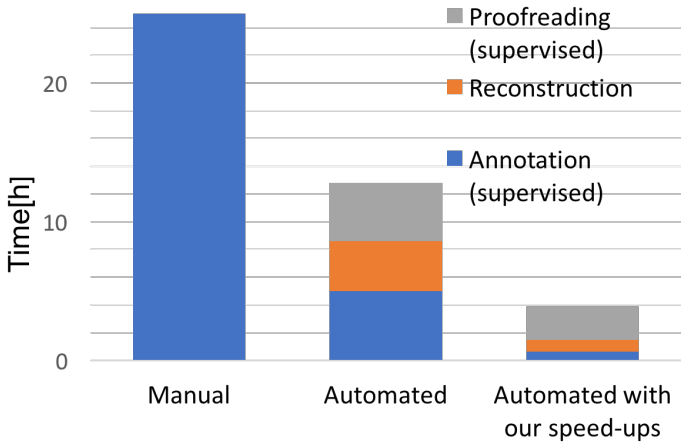


Figure 6.10 – **Time savings.** Estimated times required to reconstruct 25 out of 70 000 000 neurons in mouse brain doing all reconstruction manually (left bar), using one of the conventional automatic approaches without Active Learning and focused proofreading (middle bar) and the automatic approach with our Active Learning, proofreading and efficient formulation improvements (right bar). Our approach takes 6 times less time than manual reconstruction and requires minimal user interaction.

obtain training data, fast reconstruction and guided proofreading) takes 6 times less time when compared with manual annotation and 3 times less time than conventional automatic method (without focused Active Learning and proofreading) as shown in Fig. 6.10.

6.6 Conclusion

We have presented an attention scheme that significantly reduces the annotation effort involved both in creating training data for supervised Machine Learning and in proofreading results for delineation tasks. It does so by detecting possibly misclassified samples and considering their influence on the topology of the reconstruction. We showed that our method outperforms baselines on a variety of microscopy image stacks and can be used in interactive applications thanks to its efficient formulation. It is therefore an important step in reconstructing whole-brain neural circuitry.

7 Conclusion

In this thesis, we presented solutions to reconstructing curvilinear networks from 2D and 3D images and ways of reducing the required expert supervision. In this chapter, we first give a brief summary of this thesis and our contributions and discuss promising future research directions.

7.1 Summary

We began this thesis by presenting a great variety of curvilinear structures appearing in nature. By discussing various applications of delineation we motivated the need for automated methods suited for this problem and described challenges associated with the task. Those include image quality, great variability of scales and appearances and difficulties with obtaining labelled training data. In Chapter 2 we presented previous work in this field. We then proceeded with describing our contributions, which enable accurate and efficient delineation with minimal annotation effort.

The first step towards this goal is the training topology-aware loss that can be used for deep segmentation and, unlike the habitual binary cross-entropy, is capable of capturing higher-level statistical differences between the prediction and ground-truth. We showed that it is particularly suited for de-noising segmentation by removing unstructured background noise, as well as making the prediction smoother and more structured. Additionally, we apply an iterative refinement scheme, which progressively improves the segmentation mask.

However, segmentation itself is often not informative enough for the full delineation and we therefore aimed at recovering the full connectivity of the curvilinear network. The current approaches treat it as a post-processing step detached from the segmentation and they are often based on a set of hand-desgined heuristics. In Chapter 4 of this thesis, we described an approach that simultaneously performs segmentation as well as classification of possible

network connections. We showed that those two tasks share similar features and may benefit from one another, producing the network connectivity map without cumbersome hyper-parameter setting and complex multi-step pipeline.

Described methods are based on Machine Learning and as such they require a considerable amounts of annotated training data, which is particularly difficult to obtain in biomedical domain and for 3D image stacks. We therefore proposed two methods, which reduce the need for manual annotations. Firstly, in Chapter 5 we proposed an Active Learning approach to selecting training samples worth labelling. Unlike previous approaches, it takes into account geometric properties of curvilinear structures to devise the most informative samples by considering the ones that stand out from their neighbourhood. It allowed us to decrease the annotation effort by up to 65 %.

In Chapter 6, we introduced an alternative way to reducing annotation effort by considering samples that are particularly important from the global point of view. It works in two settings: in the Active Learning mode, it allows to efficiently label training data. In the proofreading mode, it identifies parts of the reconstruction that may be erroneous and the user should therefore focus their attention on them. We also reformulated the Mixed Integer Program formulation from [109], so that our method can find the optimal reconstruction even when dealing with very big graphs.

The presented methods help analyze vast amounts of various data, starting from microscopy brain stacks to aerial images and they are a significant step towards the full automation of this process.

7.2 Future Work

In this section we outline possible extensions to the work described in this thesis.

7.2.1 End-to-end Graph Learning

Currently, the majority of graph-based reconstruction methods, including the ones described here, require at least two steps to extract the graph. An alternative idea is to obtain graph directly from the image, without computing any segmentation. However, this is not an easy task because to the best of our knowledge, there does not exist a type of network that can take image input and output its graphical representation. The first step towards this goal was a method of [8], where road networks are traced point by point, mimicking manual annotation. An input to the network is an image patch centered at the current position and at each inference step the decision is made in which direction to proceed. A separate network is

trained to detect where tracing should stop.

However, the loss function used in that case promoted local greedy decisions about the next location. This means that even very small mistakes (*e.g.* slight deviations from the exact centerline), which may not be very important for the final reconstruction, are penalized. The network also does not learn how to go back to the true centerline after it diverts from it. An alternative solution is to use the Reinforcement Learning framework, where the algorithm has freedom to explore the image and take the right decision in places, which are especially crucial for the global reconstruction such as junctions. This approach would ensure that big mistakes that lead to significant divergence from the original trace would be penalized much more than small variations.

7.2.2 Learning Correct Topology

In Chapter 3 the features used to compute the topology loss were extracted using a neural network pre-trained on a large dataset of natural images. The intuition behind it was that the filters in lower layers of such network should capture general image properties such as edges, scale, orientation and smoothness. However, not all layers may be equally important for topology. One way to reflect it is to use a weighted topology loss, where the differences in more prominent feature maps would be penalized more. The corresponding weights can be learned directly from the data using an adversarial strategy; at one iteration, the segmentation prediction is computed and weights of segmentation network are updated according to the topology loss, which we aim to minimize. During the next iteration the weights of segmentation network are kept frozen and one updates weights corresponding to feature maps in order to maximize the loss. This way, the features that are the most sensitive to mistakes are assigned the largest weights in the subsequent iterations.

Another issue comes from the fact that the network was pre-trained on images coming from a different domain than that of binary segmentation masks. A more specified approach would be to learn features specific to curvilinear structures from the data. This could be realized by training a conditional Generative Adversarial Network, where the segmentation network acts as a generator and a discriminator is trained to distinguish predictions from ground-truth. This way the network learns the true features characterizing shapes of curvilinear structures. Similar approach was presented in [32] to detect lanes by autonomous vehicles. It could now be applied to more complex structures such as neurons or road networks.

Another possible extension in this direction could be derived from the field of mathematical topology, which studies ways of describing shapes. A recent work by Kanari *et al.* [47] introduced Topological Morphology Descriptor that was used to compare neuron morphologies and cluster them into several classes of similar morphology. Such tool could be a more princi-

pled way of comparing automatic reconstructions and ground-truth, but could also be used during the training phase to impose topologically-consistent constraints on the predictions.

7.2.3 Multimodal Data for Segmentation

In this thesis we only considered greyscale or RGB images/3D-stacks. However, in many domains it is possible to obtain additional data or information from other sources that can enhance the predictions.

One example of additional data are multi/hyperspectral images that are used in remote sensing for classifying types of vegetation. Those images may not only improve the road detection, but they can be additionally used to determine the surface condition. Moreover, OpenStreetMap provides a large database of road traces that can be aligned GoogleMaps satellite images and used to increase the size of training dataset. However, the traces are not perfect — they may miss new streets or label others inconsistently. To aid this problem one could combine images containing geolocation with GPS traces to improve the maps. A growing number of autonomous cars on the streets will allow one to collect huge amounts of ground-level footage from cameras mounted on the cars often coupled with location. Combining those two sources *and* aerial images can potentially enhance segmentation of the latter by introducing high resolution details normally not visible in satellite imagery. An example of such approach, a semantic segmentation followed by joint alignment of aerial and ground-level images, was already shown in [66] to be useful for determining the number of lanes. This is otherwise a very challenging task when using only aerial images.

In biomedical domain, it is not trivial to obtain additional training data without more effort involved in image acquisition or annotation. However, one way of producing more microscopy stacks with ready labels can be accomplished by utilizing synthetic neuron models and generating their realistic images. Such approach was already used in [4] to render neurons in fluorescent microscopy images by applying the physical principles governing microscopic image formation. It could be useful to further expand the training set described in this thesis.

A An appendix

A.1 DIADEM score

DIADEM score [7] was originally developed to evaluate the neuron reconstruction but it can be used to assess any tree-like structures. In Chapter 6 of this thesis it was used to measure the topological change in reconstruction after alterign one of the edge's weight. It measures the topological similarity between two trees which are usually a ground-truth and an automatically produced delineation. Every node in the ground-truth is assigned matches i.e. points in the delineation that are within a certain distance from the given ground-truth node. To assess the topology, for every match the computed tree is traversed up to the match's ancestor bifurcation. This procedure is repeated until the ancestor of target node and the ancestor of its match correspond to each other (are within certain distance). Once they are established, the error is computed as follows:

$$E = \frac{|L_G - L_R|}{L_G} \tag{A.1}$$

where L_G and L_R are the lengths of traversed paths in ground-truth and reconstruction respectively as shown in Fig. A.1. If the error is less than a threshold, then the match is considered to be true. Moreover, DIADEM score introduces the notion of misses i.e. leaf nodes in the ground-truth that do not have a match in the reconstruction and excess nodes i.e. terminal vertices in the delineation that do not have a corresponding match in the ground-truth within a certain distance. The importance of each node is weighted by its degree. Once all the vertices in the ground-truth are processed, the weights of those that were truly matched are added to the numerator. The total weight of all nodes in ground-truth and the weight of excess nodes are added in the denominator. The final DIADEM score is taken to be the ratio between those two quantities.

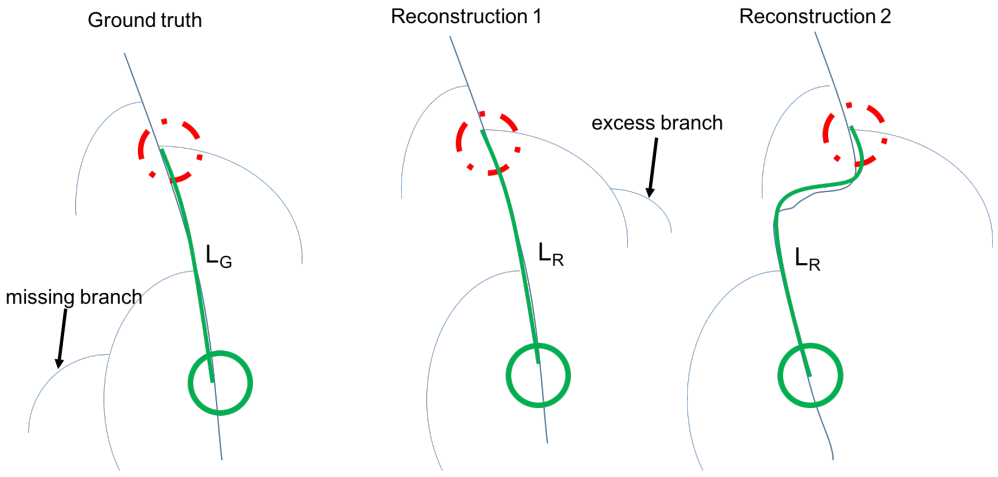


Figure A.1 – **DIADEM score.** An example of ground-truth and two delineations. Green circles represent matches and red circles their ancestors. L_G and L_R are the distances between the matches and ancestors. In this case, the error E of reconstruction 2 will be higher than the one for reconstruction 1.

Bibliography

[1] Humanity & Inclusion. https://hi.org/en/index.snc. Accessed: 2018-10-04.

[2] Introducing Sentinel-1. https://www.esa.int/Our_Activities/Observing_the_Earth/Copernicus/Sentinel-1/Introducing_Sentinel-1. Accessed: 2018-10-04.

[3] Mapping Challenge: Building Maps with Machine Learning. https://www.crowdai.org/challenges/mapping-challenge. Accessed: 2018-10-04.

[4] M. Abdellah, A. Bilgili, S. Eilemann, H. Markram, and F. Schuermann. A Computational Model of Light-Sheet Fluorescence Microscopy using Physically-based Rendering. The Eurographics Association, 2015.

[5] F. Aguet, M. Jacob, and M. Unser. Three-Dimensional Feature Detection Using Optimal Steerable Filters. In *International Conference on Image Processing*, September 2005.

[6] I. Arganda-Carreras, S. Turaga, D. Berger, D. Ciresan, A. Giusti, L. Gambardella, J. Schmidhuber, D. Laptev, S. Dwivedi, J. Buhmann, T. Liu, M. Seyedhosseini, T. Tasdizen, L. Kamentsky, R. Burget, V. Uher, X. Tan, C. Sun, T. Pham, E. Bas, M. Uzunbas, A. Cardona, J. Schindelin, and S. Seung. Crowdsourcing the Creation of Image Segmentation Algorithms for Connectomics. *Frontiers in Neuroanatomy*, page 142, 2015.

[7] G. Ascoli, K. Svoboda, and Y. Liu. Digital Reconstruction of Axonal and Dendritic Morphology DIADEM Challenge, 2010.

[8] F. Bastani, S. He, M. Alizadeh, H. Balakrishnan, S. Madden, S. Chawla, S. Abbar, and D. Dewitt. Roadtracer: Automatic Extraction of Road Networks from Aerial Images. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[9] C. Becker, R. Rigamonti, V. Lepetit, and P. Fua. Supervised Feature Learning for Curvilinear Structure Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, September 2013.

[10] C. Blum and B. Calvo. A Matheuristic for the Minimum Weight Rooted Arborescence Problem. *Journal of Heuristics*, 21(4):479–499, 2015.

[11] D. Breitenreicher, M. Sofka, S. Britzen, and S. Zhou. Hierarchical Discriminative Framework for Detecting Tubular Structures in 3D Images. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2013.

[12] D. Chai, W. Forstner, and F. Lafarge. Recovering Line-Networks in Images by Junction-Point Processes. In *Conference on Computer Vision and Pattern Recognition*, 2013.

[13] J. Chen, Y. Sato, and S. Tamura. Orientation Space Filtering for Multiple Orientation Line Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):417–429, 2000.

[14] L. Chen, G. Papandreou, I.Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. 2018.

[15] G. Cheng, Y. Wang, S. Xu, H. Wang, S. Xiang, and C. Pan. Automatic Road Detection and Centerline Extraction via Cascaded End-To-End Convolutional Neural Network. *IEEE Trans. Geoscience and Remote Sensing*, 55(6):3322–3337, 2017.

[16] Ö. Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 424–432, 2016.

[17] D. Cohn, Z. Ghahramani, and M. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[18] I. Dagan and S. P. Engelson. Committee-Based Sampling for Training Probabilistic Classifiers. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[19] A. Darpe. A novel way to detect transverse surface crack in a rotating shaft. *Journal of Sound and Vibration*, 305(1):151–171, 2007.

[20] D. D. D.D. Lewis and W. Gale. A Sequential Algorithm for Training Text Classifiers. In *ACM SIGIR proceedings on Research and Development in Information Retrieval*, 1994.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[22] V. Dercksen, H. Hege, and M. Oberlaender. The Filament Editor: An Interactive Software Environment for Visualization, Proof-Editing and Analysis of 3D Neuron Morphology. *Neuroinformatics*, 12:325–339, 2014.

[23] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[24] S. Ebert, M. Fritz, and B. Schiele. RALF: A Reinforced Active Learning Formulation for Object Class Recognition. In *Conference on Computer Vision and Pattern Recognition*, 2012.

[25] A. V. Etten. SpaceNet Road Detection and Routing Challenge Part II — APLS Implementation. https://medium.com/the-downlinq/

spacenet-road-detection-and-routing-challenge-part-ii-apls-implementation-92acd86f4094. Accessed: 2018-12-09.

[26] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (Voc) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[27] A. Fathi, A. Farhadi, and J. Rehg. Understanding Egocentric Activities. In *International Conference on Computer Vision*, pages 407–414, 2011.

[28] A. Freytag, E. Rodner, and J. Denzler. Selecting Influential Examples: Active Learning with Expected Model Output Changes. In *European Conference on Computer Vision*, 2014.

[29] H. Fu, Y. Xu, D. W. K. Wong, and J. Liu. Retinal vessel segmentation via deep learning network and fully-connected conditional random fields. *International Symposium on Biomedical Imaging*, pages 698–701, 2016.

[30] Y. Ganin and V. Lempitsky. N4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms. In *Asian Conference on Computer Vision*, pages 536–551, 2014.

[31] L. A. Gatys, A. S. Ecker, and M. Bethge. Image Style Transfer Using Convolutional Neural Networks. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[32] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann. EL-GAN: Embedding Loss Driven Generative Adversarial Networks for Lane Detection. *CoRR*, 2018.

[33] P. Glowacki, M. Pinheiro, E. Turetken, R. Sznitman, D. Lebrecht, A. Holtmaat, J. Kybic, and P. Fua. Reconstructing Evolving Tree Structures in Time Lapse Sequences. In *Conference on Computer Vision and Pattern Recognition*, 2014.

[34] P. Glowacki, M. Pinheiro, E. Turetken, R. Sznitman, D. Lebrecht, A. Holtmaat, J. Kybic, and P. Fua. Reconstructing Evolving Tree Structures in Time Lapse Sequences by Enforcing Time-Consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):755–761, 2018.

[35] G. Gonzalez, F. Fleuret, and P. Fua. Automated Delineation of Dendritic Networks in Noisy Image Stacks. In *European Conference on Computer Vision*, pages 214–227, 2008.

[36] L. Gu and L. Cheng. Learning to Boost Filamentary Structure Segmentation. pages 639–647, 2015.

[37] C. Heneghan, J. Flynn, M. O'Keefe, and M. Cahill. Characterization of changes in blood vessel width and tortuosity in retinopathy of prematurity using image analysis. *Medical Image Analysis*, 6(4):407–429, 2002.

[38] X. Huang, J. Gao, L. Wang, and R. Yang. Examplar-Based Shape from Shading. In *3DIM*, pages 349–356, 2007.

[39] X. Huang and L. Zhang. Road Centreline Extraction from High-Resolution Imagery Based on Multiscale Structural Features and Support Vector Machines. *International Journal of Remote Sensing*, 30:1977–1987, 2009.

[40] J. Iglesias, E. Konukoglu, A. Montillo, Z. Tu, and A. Criminisi. Combining Generative and Discriminative Models for Semantic Segmentation. In *Information Processing in Medical Imaging*, 2011.

[41] M. Jacob and M. Unser. Design of Steerable Filters for Feature Detection Using Canny-Like Criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1007–1019, August 2004.

[42] S. D. Jain and K. Grauman. Active Image Segmentation Propagation. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[43] M. Januszewski, J. Maitin-Shepard, P. Li, J. Kornfeld, W. Denk, and V. Jain. Flood-Filling Networks. *arXiv Preprint*, 2016.

[44] H. Jelinek, M. Cree, J. Leandro, J. Soares, R. Cesar, and A. Luckie. Automated segmentation of retinal blood vessels and identification of proliferative diabetic retinopathy. 24:1448–56, 2007.

[45] S. Jeong, Y. Tarabalka, and J. Zerubia. Marked Point Process Model for Curvilinear Structures Extraction. In *Conference on Computer Vision and Pattern Recognition*, pages 436–449, 2015.

[46] P. Kaiser, J. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler. Learning Aerial Image Segmentation from Online Maps. *IEEE Transactions on Geoscience and Remote Sensing*, 55:6054–6068, 2017.

[47] L. Kanari, P. Dlotko, M. Scolamiero, R. Levi, J. Shillcock, K. Hess, and H. Markram. Quantifying Topological Invariants of Neuronal Morphologies. *arXiv Preprint*, 2016.

[48] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active Learning with Gaussian Processes for Object Categorization. In *International Conference on Computer Vision*, 2007.

[49] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimisation. In *International Conference on Learning Representations*, 2015.

[50] K. Konyushkova, R. Sznitman, and P. Fua. Introducing Geometry into Active Learning for Image Segmentation. In *International Conference on Computer Vision*, 2015.

[51] M. Koziński, A. Mosinska, M. Salzmann, and P. Fua. Learning to Segment 3D Linear Structures Using Only 2D Annotations. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 283–291, 2018.

[52] P. Krähenbühl and V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *Advances in Neural Information Processing Systems*, 2011.

[53] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.

[54] W. Lamadé, G. Glombitza, L. Fischer, P. Chiu, C. Cárdenas, M. Thorn, H. Meinzer, L. Grenacher, H. Bauer, T. Lehnert, and C. Herfarth. The impact of 3-dimensional reconstructions on operation planning in liver surgery. *Archives of Surgery*, 135:1256–61, 2000.

[55] M. Law and A. Chung. Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. In *European Conference on Computer Vision*, 2008.

[56] C. Li and K. Kitani. Pixel-Level Hand Detection in Ego-Centric Videos. In *Conference on Computer Vision and Pattern Recognition*, 2013.

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[58] T. Liu, A. Moore, A. Gray, and K. Yang. An Investigation of Practical Approximate Nearest Neighbor Algorithm. In *Advances in Neural Information Processing Systems*, 2004.

[59] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg. SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision*, 2016.

[60] C. Long, G. Hua, and A. Kapoor. Active Visual Recognition with Expertise Estimation in Crowdsourcing. In *International Conference on Computer Vision*, 2013.

[61] R. Mackowiak, P. Lenz, O. Ghori, F. Diego, O. Lange, and C. Rother. CEREALS - Cost-Effective Region-Based Active Learning for Semantic Segmentation. In *British Machine Vision Conference*, 2018.

[62] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool. Deep Retinal Image Understanding. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2016.

[63] H. Markram, K. Meier, T. Lippert, S. Grillner, R. Frackowiak, S. Dehaene, A. Knoll, H. Sompolinsky, K. Verstreken, J. Defelipe, S. Grant, J. Changeux, and A. Saria. Introducing the Human Brain Project. *Procedia Computer Science*, 7:39–42, 2011.

[64] H. Marquering, J. Dijkstra, P. de Koning, B. Stoel, and J. Reiber. Towards quantitative Analysis of Coronary CTA. *The International Journal of Cardiovascular Imaging*, 21:73–84, 2005.

[65] G. Mattyus, S. Wang, S. Fidler, and R. Urtasun. Enhancing Road Maps by Parsing Aerial Images Around the World. In *International Conference on Computer Vision*, pages 1689–1697, 2015.

[66] G. Mattyus, S. Wang, S. Fidler, and R. Urtasun. HD Maps: Fine-Grained Road Segmentation by Parsing Ground and Aerial Images. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[67] G. Mattyusand, W. L. R., and Urtasun. Deep Roadmapper: Extracting Road Topology from Aerial Images. In *International Conference on Computer Vision*, pages 3438–3446, 2017.

[68] J. Merkow, A. Marsden, D. Kriegman, and Z. Tu. Dense Volume-to-Volume Vascular Boundary Detection. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 371–379, 2016.

[69] V. Mnih. *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto, 2013.

[70] V. Mnih and G. Hinton. Learning to Detect Roads in High-Resolution Aerial Images. In *European Conference on Computer Vision*, pages 210–223, 2010.

[71] V. Mnih and G. Hinton. Learning to Label Aerial Images from Noisy Data. In *International Conference on Machine Learning*, 2012.

[72] J. Montoya-Zegarra, J. Wegner, L. Ladicky, and K. Schindler. Mind the Gap: Modeling Local and Global Context in (Road) Networks. In *German Conference on Pattern Recognition*, 2014.

[73] A. Mosinska, P. Marquez-neila, M. Kozinski, and P. Fua. Beyond the Pixel-Wise Loss for Topology-Aware Delineation. In *Conference on Computer Vision and Pattern Recognition*, 2018.

[74] A. Mosinska, R. Sznitman, P. Glowacki, and P. Fua. Active Learning for Delineation of Curvilinear Structures. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[75] A. Mosinska, K. Tarnawski, and P. Fua. Active Learning and Proofreading for Delineation of Curvilinear Structures. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 165–173, September 2017.

[76] A. Narayanaswamy, Y. Wang, and B. Roysam. 3D Image Pre-Processing Algorithms for Improved Automated Tracing of Neuronal Arbors. *Neuroinformatics*, 9(2-3):219–231, 2011.

[77] P. F. Neher, M. Götz, T. Norajitra, C. Weber, and K. H. Maier-Hein. A Machine Learning Based Approach to Fiber Tractography Using Classifier Voting. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 45–52, 2015.

[78] A. Newell, K. Yang, and J. Deng. Stacked Hourglass Networks for Human Pose Estimation. In *European Conference on Computer Vision*, 2016.

[79] A. Niculescu-mizil and R. Caruana. Obtaining Calibrated Probabilities from Boosting. *CoRR*, abs/1207.1403, 2012.

[80] J. Orlando and M. Blaschko. Learning Fully-Connected CRFs for Blood Vessel Segmentation in Retinal Images. In *Conference on Medical Image Computing and Computer Assisted Intervention*, 2014.

[81] E. Oveisi, A. Letouzey, D. Alexander, Q. Jeangros, R. Schaublin, G. Lucas, P. Fua, and C. Hebert. Tilt-Less 3D Electron Imaging and Reconstruction of Complex Curvilinear Structures. *Nature Scientific Reports*, 7(10630), 2017.

[82] E. Oveisi, A. Letouzey, S. D. Zanet, G. Lucas., M. Antoni, P. Fua, and C. Hebert. Stereo-Vision Three-Dimensional Reconstruction of Curvilinear Structures Imaged with a TEM. *Ultra Microscopy*, 184(A), 2017.

[83] S. Park, S. Ahmad, C.-B. Yun, and Y. Roh. Multiple Crack Detection of Concrete Structures Using Impedance-based Structural Health Monitoring Techniques. *Experimental Mechanics*, 46(5):609–618.

[84] H. Peng, F. Long, and G. Myers. Automatic 3D Neuron Tracing Using All-Path Pruning. *Bioinformatics*, 27(13):239–247, 2011.

[85] H. Peng, F. Long, T. Zhao, and E. Myers. Proof-Editing is the Bottleneck of 3D Neuron Reconstruction: the Problem and Solutions. *Neuroinformatics*, 9(2):103–105, 2011.

[86] P. Pinheiro and R. Collobert. Recurrent Neural Networks for Scene Labelling. In *International Conference on Machine Learning*, 2014.

[87] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[88] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 234–241, 2015.

[89] D. Salazar. Satellite Photos Help Indonesian Relief Workers Sort Through Earthquake, Tsunami Aftermath. https://www.space.com/42005-indonesia-tsunami-earthquake-satellite-imagery-relief.html. Accessed: 2018-10-04.

[90] A. Santamaría-Pang, P. Hernandez-Herrera, M. Papadakis, P. Saggau, and I. Kakadiaris. Automatic Morphological Reconstruction of Neurons from Multiphoton and Confocal Microscopy Images Using 3D Tubular Models. *Neuroinformatics*, pages 1–24, 2015.

[91] G. Schohn and D. Cohn. Less is More: Active Learning with Support Vector Machines. In *International Conference on Machine Learning*, 2000.

[92] B. Settles. Active Learning Literature Survey. Technical report, University of Wisconsin–Madison, 2010.

[93] B. Settles. From Theories to Queries : Active Learning in Practice. *Active Learning and Experimental Design*, 2011.

[94] B. Settles and M. Craven. Active Learning with Real Annotation Costs. In *Advances in Neural Information Processing Systems*, pages 1–10, 2008.

[95] B. Settles and M. Craven. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

[96] M. Seyedhosseini, M. Sajjadi, and T. Tasdizen. Image Segmentation with Cascaded Hierarchical Models and Logistic Disjunctive Normal Networks. In *International Conference on Computer Vision*, 2013.

[97] H. Shen. Neuron encyclopaedia fires up to reveal brain secrets. https://www.nature.com/news/neuron-encyclopaedia-fires-up-to-reveal-brain-secrets-1.17232. Accessed: 2018-10-04.

[98] W. Shen, B. Wang, Y. Jiang, Y. Wang, and A. Yuille. Multi-stage Multi-recursive-input Fully Convolutional Networks for Neuronal Boundary Detection. In *International Conference on Computer Vision*, 2017.

[99] B. Siddiquie and A. Gupta. Beyond Active Noun Tagging: Modeling Contextual Interactions for Multi-Class Active Learning. In *Conference on Computer Vision and Pattern Recognition*, 2010.

[100] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*, 2015.

[101] A. Sironi, V. Lepetit, and P. Fua. Projection Onto the Manifold of Elongated Structures for Accurate Extraction. In *International Conference on Computer Vision*, 2015.

[102] A. Sironi, E. Turetken, V. Lepetit, and P. Fua. Multiscale Centerline Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1327–1341, 2016.

[103] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. van Ginneken. Ridge Based Vessel Segmentation in Color Images of the Retina. *IEEE Transactions on Medical Imaging*, 2004.

[104] C. Steger. An Unbiased Detector of Curvilinear Structures. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, pages 113–125, 1998.

[105] R. Stoica, X. Descombes, and J. Zerubia. A Gibbs Point Process for Road Extraction from Remotely Sensed Images. *International Journal of Computer Vision*, 57(2):121–136, 2004.

[106] K. Sun, N. Sang, and T. Zhang. Marked Point Process for Vascular Tree Extraction on Angiogram. In *Conference on Computer Vision and Pattern Recognition*, pages 467–478, 2007.

[107] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Machine Learning*, 2002.

[108] E. Turetken, C. Becker, P. Glowacki, F. Benmansour, and P. Fua. Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery Using Multi-Directional Oriented Flux. In *International Conference on Computer Vision*, December 2013.

[109] E. Turetken, F. Benmansour, B. Andres, P. Glowacki, H. Pfister, and P. Fua. Reconstructing Curvilinear Networks Using Path Classifiers and Integer Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(12):2515–2530, 2016.

[110] E. Turetken, F. Benmansour, B. Andres, H. Pfister, and P. Fua. Reconstructing Loopy Curvilinear Structures Using Integer Programming. In *Conference on Computer Vision and Pattern Recognition*, June 2013.

[111] E. Turetken, F. Benmansour, and P. Fua. Automated Reconstruction of Tree Structures Using Path Classifiers and Mixed Integer Programming. In *Conference on Computer Vision and Pattern Recognition*, June 2012.

[112] C. Ventura, J. Pont-Tuset, S. Caelles, K. Maninis, and L. V. Gool. Iterative Deep Learning for Network Topology Extraction. In *British Machine Vision Conference*, 2018.

[113] A. Vezhnevets, J. Buhmann, and V. Ferrari. Active Learning for Semantic Segmentation with Expected Change. In *Conference on Computer Vision and Pattern Recognition*, 2012.

[114] S. Vijayanarasimhan and K. Grauman. What's It Going to Cost You?: Predicting Effort Vs. Informativeness for Multi-Label Image Annotations. In *Conference on Computer Vision and Pattern Recognition*, pages 2262–2269, 2009.

[115] J. Wegner, J. Montoya-Zegarra, and K. Schindler. A Higher-Order CRF Model for Road Network Extraction. In *Conference on Computer Vision and Pattern Recognition*, pages 1698–1705, 2013.

[116] J. Wegner, J. Montoya-Zegarra, and K. Schindler. Road Networks as Collections of Minimum Cost Paths. *International Society for Photogrammetry and Remote Sensing*, 108:128–137, 2015.

[117] C. Wiedemann, C. Heipke, H. Mayer, and O. Jamet. Empirical Evaluation of Automatically Extracted Road Axes. In *Empirical Evaluation Techniques in Computer Vision*, pages 172–187, 1998.

[118] S. Xie and Z. Tu. Holistically-Nested Edge Detection. *International Conference on Computer Vision*, pages 1395–1403, 2015.

[119] J. Yang, M. Hao, X. Liu, Z. Wan, N. Zhong, and H. Peng. FMST: an Automatic Neuron Tracing Method Based on Fast Marching and Minimum Spanning Tree. *Neuroinformatics*, 2018.

[120] J. A. M. Zegarra, J. D. Wegner, L. Ladicky, and K. Schindler. Mind the Gap: Modeling Local and Global Context in (Road) Networks. In *Pattern Recognition*, 2014.

[121] Z. Zhang, Q. Liu, and Y. Wang. Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, 15:749–753, 2018.

[122] D. Zheng, K. M. Yi, Y. Hu, F. Wang, P. Fua, and M. Salzmann. Eigendecomposition-Free Training of Deep Networks with Zero Eigenvalue-Based Losses. In *European Conference on Computer Vision*, 2018.

[123] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with Local and Global Consistency. In *Advances in Neural Information Processing Systems*, pages 321–328, 2004.

[124] S. K. Zhou, C. Tietjen, G. Soza, A. Wimmer, C. Lu, Z. Puskas, D. Liu, and D. Wu. A Learning Based Deformable Template Matching Method for Automatic Rib Centerline Extraction and Labeling in CT Images. In *Conference on Computer Vision and Pattern Recognition*, 2012.

[125] Z. Zhou, K. Hsien-Chi, H. Peng, and F. Long. DeepNeuron: an open deep learning toolbox for neuron tracing. *Brain Informatics*, 5, 2018.

[126] X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer. Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. 5:8–36, 2017.

[127] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang. Cracktree: Automatic Crack Detection from Pavement Images. *Pattern Recognition Letters*, 33(3):227–238, 2012.

AGATA MOSINSKA

Av. de l'Eglise-Anglaise 14, 1006 Lausanne
+41 764 140 136

aj.mosinska@gmail.com

linkedin.com/in/agatamosinska

Strengths

Computer Vision • Image Segmentation • Machine Learning • Convolutional Neural Networks • Algorithm Design • Data Analysis • Research • Python • PyTorch

Professional experience

2014- present **CVLab, École Polytechnique Fédérale de Lausanne (EPFL)** **Doctoral Assistant**

- Developed a learning approach for an automatic segmentation of curvilinear structures
- Designed two Active Learning methods that reduce the annotation effort by up to 80%
- Collaborated intensively with colleagues on multiple projects and with scientists from various fields (Neuroscience, Physics) to develop tools that facilitate their research
- Presented work during top tier conferences (both posters and oral presentations)
- Led a team of Teaching Assistants responsible for designing Computer Vision practicals, exams and holding consultations for 90 Master students
- Supervised 3 Master students during their semester projects

2014 **Skin Analytics, London, UK** **Image Processing Researcher**

- Designed and extended image analysis technology to track skin changes associated with various skin diseases during 6-months part-time job
- Identified promising research techniques for dry skin assessment, evaluated their applicability and implemented the most promising methods
- Presented the current technology and the possible extensions to the potential customers

2013 **QUALCOMM Research and Development, Cambridge, UK** **R&D Intern**

- Worked for 3-months in a 12-people team on a gesture-controlled media centre
- Conducted a comparison of *hand skeletonisers* by implementing a testing tool in C++ and testing technology available on the market

2012 **SONY Broadcast & Professional Research Labs, Basingstoke, UK** **R&D Intern**

- Researched the principles of real-time enhancement of surgical videos
- Advised and implemented a novel technique of edge detection and noise recognition in C++

Education

2014-2018 (expected) **École Polytechnique Fédérale de Lausanne (EPFL), Ph.D. Computer Science**, **Computer Vision Laboratory,** thesis director - Professor Pascal Fua

Specialization: *Detection of Curvilinear Structures* – developed Machine Learning techniques for delineation of tubular networks with broad application (roads, cracks, blood vessels, neurons)

2010-2014 **Imperial College London**, **M.Eng Biomedical Engineering** - high 1st class degree

Specialization: Electrical Engineering

2007-2010 **International Baccalaureate Programme, Poznan, Poland**

Secondary school diploma

Specialization – Mathematics, Physics, Chemistry

Other activities

2018 -present **Human Brain Project** **Student Ambassador**

- Worked as a part of 6-people team organizing 2nd HBP Student Conference; responsible for preparing scientific programme, reviewing abstracts and awarding the prizes
- Promoted events associated with HBP Education Programme (conferences, workshops)

2009-2013 **Mentoring for secondary and high school students** **Tutor**

- Explained mathematical and physical concepts in a concise way
- Motivated students to attempt problems themselves
- Tailored approach to individual's strengths and weaknesses

2011 **Imperial College London** **Summer Intern**

- Developed virtual mechanics labs during 10-weeks full-time internship
- Advised possible improvements in *SimMechanics* and its documentation

Technical skills

Programming	Python, C, C++, Matlab
Software development	Agile methodologies, version control (Git,SVN), debugging, software maintenance
Operating systems	macOS, Linux, MS Windows
Other skills	Image Processing, Signal Processing, Convex Optimization, Mathematical Modelling

Selected publications

Full list

2018 **Mosinska A**. et al., *Beyond the Pixel-Wise Loss for Topology-Aware Delineation*, CVPR 2018

2017 **Mosinska A**. et al., *Active Learning and Proofreading for Delineation of Curvilinear Structures*, MICCAI 2017 (selected for an oral presentation)

2016 **Mosinska A**. et al*., Active Leaning for Delineation of Curvilinear Structures*, CVPR 2016

Achievements

2018	Winner of Zeiss Imaging Challenge, invited to present skin tracking app at Zeiss headquarter
2015	Winner of the Human Brain Project Writing Competition, invited talk during HBP Summit
2014	Full Fellowship from Computer and Communication Sciences Doctoral School
2011-2014	Placed on the Engineering Dean's List for an excellent academic record (top 5% of class)
2007	Awarded scholarship as one of 10 students in Poland for outstanding achievements in sciences

Languages

English (C2) • Polish (native) • German (B1) • French (A2)

Interests

Mountaineering	Trips to Alps, Tatras and Andes including 5 000 m summits
Bike touring	Multiday touring trips across Europe
Yoga	A regular practitioner of Barkan-style yoga

Personal information

Date of Birth	03.11.1991
Nationality	Polish (Swiss work permit B)
Driving licence	Category B

**Ce document a été imprimé au Centre d'impression EPFL,
imprimerie climatiquement neutre, certifiée myClimate.**