

Topologically Better Delineation of Curvilinear Structures

Présentée le 14 décembre 2023

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

Doruk ONER

Acceptée sur proposition du jury

Dr M. Rajman, président du jury
Prof. P. Fua, directeur de thèse
Dr J. Funke, rapporteur
Dr M. Januszewski, rapporteur
Prof. C. Petersen, rapporteur

Mum olmak kolay değildir.
Işık saçmak için önce yanmak gerek.
— Rumi

To my family...

Acknowledgements

I would like to express my heartfelt gratitude to the following individuals and groups who have played a significant role in the completion of this thesis.

First and foremost, I extend my deepest appreciation to my supervisor, Professor Pascal Fua. Pascal's guidance, expertise, and unwavering support throughout this research journey have been invaluable. His insightful feedback, patience, and motivation have pushed me to explore new horizons and strive for excellence. I am truly grateful for the opportunity to learn under his mentorship.

I am also indebted to the members of my thesis committee, Professor Carl Petersen, Doctor Jan Funke, and Doctor Michał Januszewski and jury president Doctor Martin Rajman for their valuable insights, constructive criticisms, and suggestions that have enhanced the quality of this work.

My sincere thanks go to my collaborators, both within and outside the academic community. Their willingness to share their knowledge, collaborate on experiments, and exchange ideas has broadened my perspectives and enriched my research experience. Their contributions have been crucial in driving this project forward.

I would like to acknowledge the members of CVLab. Their support and intellectual discussions have created an inspiring environment for me to grow as a researcher. I am grateful for the shared experiences, scientific debates, and memorable moments we have had together. Their friendship has made this journey all the more enjoyable. Special thanks to Mateusz Kozinski and Leonardo Citraro with whom I shared an office for the past 4 years. They not only represent excellent collaborators but also supportive friends. And to Vedit, who has always been an understanding friend.

I would like to thank my dear friends for standing by me, providing encouragement, and understanding during the challenging times of my PhD journey. Their presence, laughter, and words of motivation have been a source of strength throughout this process. In particular: To Sinem, my amazing companion throughout the entire Ph.D. journey, for always being there for

Acknowledgements

me, facing challenges together, and creating unforgettable memories and providing feedback even for this comment; To Duru, for deep conversations that last hours and unforgettable vacations that will never end; To Musto, for his guidance and sharing his life experiences. Also for sharing the darkest times over saz, backgammon, and his art; To Sinan and Murat, for the fun times and for bringing joy and laughter to my life in Lausanne; To Martina and Diana, your presence and encouragement in the final part of my Ph.D. turned tough moments into beautiful ones, and I am thankful for the positivity and warmth you brought into my life.

Last but certainly not least, I want to express my deepest gratitude to my family. Their unwavering love, belief in my abilities, and constant support have been the foundation of my achievements. Their sacrifices, encouragement, and understanding have fueled my determination to pursue my academic goals. I am eternally grateful for everything they have done for me.

To everyone who has contributed, directly or indirectly, to the completion of this thesis, I extend my heartfelt appreciation. Their support has been indispensable in shaping this research endeavor.

Lausanne, July 19, 2023

Doruk Öner

Abstract

Curvilinear structures are frequently observed in a variety of domains and are essential for comprehending neural circuits, detecting fractures in materials, and determining road and irrigation canal networks. It can be costly and time-consuming to manually extract these structures, so automated systems are required for faster and more accurate analysis. Recently deep learning-based approaches that rely solely on deep neural networks are being used for automatic delineation of such structures. However, preserving the topology of the curvilinear structures, which is essential for downstream applications, is a significant challenge. Furthermore, deep learning models require vast quantities of precisely annotated data, which is difficult to acquire, especially for 3D data. Commonly used pixel-wise loss functions cannot capture the topology of curvilinear structures and deep networks that are trained with such losses are prone to imprecisions in the annotations. In this thesis, we propose topology-aware loss functions to tackle these problems and improve the topology of the reconstructions.

We begin by introducing a connectivity-oriented loss function for extracting network-like structures from 2D images. We express the connectivity of curvilinear structures in terms of disconnections that they create between background regions of the image. Our loss function is designed to prevent such unwanted connections between background regions, and therefore close the gaps in predicted structures. Then, we focus on using Persistent Homology (PH) to improve the topological quality of the reconstructions. We propose a new filtration technique by fusing two existing approaches: filtration by thresholding and height function. With the proposed technique, we include location information of topological errors and increase the descriptive power of PH. In order to tackle imprecise annotations, we propose an active contour model based loss function. We treat annotations as contour models and allow them to deform themselves over correct centerlines during training while preserving the topology of the structure. Our final contribution is extending the aforementioned connectivity-oriented loss function to work with 3D data as well. We achieve this by computing the loss on 2D projections. With this method, we also reduce the annotation effort required to provide training data. We demonstrate, in experiments on 2D (satellite images of roads and irrigation canals) and 3D (Magnetic Resonance Angiography, Two-Photon Microscopy) datasets, that our loss functions significantly improve the topological quality of the reconstructions.

Abstract

Keywords: Computer Vision, Deep Learning, Delineation, Curvilinear Structures, Connectivity, Topology.

Zusammenfassung

Kurvilineare Strukturen werden häufig in verschiedenen Bereichen beobachtet und sind wesentlich für das Verständnis neuronaler Schaltkreise, der Erkennung von Brüchen in Materialien sowie der Bestimmung von Straßen- und Bewässerungskanalsystemen. Es kann teuer und zeitaufwändig sein, diese Strukturen manuell zu extrahieren, daher sind automatisierte Systeme erforderlich, um eine schnellere und genauere Analyse zu ermöglichen. In letzter Zeit werden Deep-Learning-basierte Ansätze, die ausschließlich auf tiefen neuronalen Netzwerken basieren, für die automatische Abgrenzung solcher Strukturen verwendet. Die Erhaltung der Topologie der kurvilinearen Strukturen, die für nachgelagerte Anwendungen wesentlich ist, stellt jedoch eine große Herausforderung dar. Ferner benötigen Deep-Learning-Modelle große Mengen präzise annotierter Daten, die insbesondere für 3D-Daten schwer zu beschaffen sind. Häufig verwendete pixelweise Verlustfunktionen können die Topologie von kurvilinearen Strukturen nicht erfassen, und tiefe Netzwerke, die mit solchen Verlusten trainiert werden, neigen zu Ungenauigkeiten in den Annotationen. In dieser Arbeit schlagen wir topologiebewusste Verlustfunktionen vor, um diese Probleme zu lösen und die Topologie der Rekonstruktionen zu verbessern. Wir fangen damit an, eine verbindungsorientierte Verlustfunktion zur Extraktion netzwerkähnlicher Strukturen aus 2D-Bildern einzuführen. Wir drücken die Verbindungen von kurvilinearen Strukturen in Bezug auf Unterbrechungen aus, die sie zwischen Hintergrundregionen des Bildes erzeugen. Unsere Verlustfunktion ist darauf ausgelegt, diese unerwünschten Verbindungen zwischen Hintergrundregionen zu verhindern und die Lücken in den vorhergesagten Strukturen zu schließen. Anschließend konzentrieren wir uns darauf, die topologische Qualität der Rekonstruktionen mithilfe der Persistent Homology (PH) zu verbessern. Wir schlagen eine neue Filtrationstechnik vor, indem wir zwei bestehende Ansätze kombinieren: Filtration durch Schwellenwertbildung und Höhenfunktion. Mit der vorgeschlagenen Technik erfassen wir auch die Positionsinformationen der topologischen Fehler und erhöhen die beschreibende Kraft der PH. Um ungenaue Annotationen zu bewältigen, schlagen wir eine Verlustfunktion basierend auf einem aktiven Konturmodell vor. Wir behandeln Annotationen als Konturmodelle und erlauben ihnen, sich während des Trainings entlang korrekter Mittellinien zu verformen, wobei die Topologie der Struktur erhalten bleibt. Unser letzter Beitrag besteht darin, die zuvor erwähnte verbindungsorientierte Verlustfunktion auch für 3D-Daten zu erweitern. Dies erreichen wir, indem wir den Verlust auf 2D-Projektionen berechnen. Mit dieser Methode reduzieren wir auch den Annotationsaufwand zur Bereitstellung von Trainingsdaten. In Experimenten mit 2D-Datensätzen (Satellitenbilder von Straßen und Bewässerungskanälen) und 3D-Datensätzen (Magnetresonanztomographie, Zweiphotonenmikroskopie) zeigen

Zusammenfassung

wir, dass unsere Verlustfunktionen die topologische Qualität der Rekonstruktionen deutlich verbessern.

Schlüsselwörter: Computer Vision, Deep Learning, Abgrenzung, Kurvilinearen Strukturen, Konnektivität, Topologie

Résumé

Les structures curvilignes sont fréquemment observées dans divers domaines et sont essentielles pour comprendre les circuits neuronaux, détecter les fissures dans les matériaux, et identifier les réseaux routiers et de canaux d'irrigation. L'extraction manuelle de ces structures peut être coûteuse et chronophage, d'où la nécessité de systèmes automatisés pour une analyse plus rapide et plus précise. Récemment, des approches basées sur l'apprentissage profond qui reposent uniquement sur des réseaux de neurones artificiels profonds sont utilisées pour la délimitation automatique de telles structures. Cependant, préserver la topologie des structures curvilignes, qui est essentielle pour les applications en aval, est un défi important. De plus, les modèles d'apprentissage profond nécessitent d'énormes quantités de données précisément annotées, ce qui est difficile à obtenir, surtout pour les données en 3D. Les fonctions de coût basées sur les pixels couramment utilisés ne peuvent pas capturer la topologie des structures curvilignes et les réseaux profonds qui sont entraînés avec cette forme de supervision sont sujets à des imprécisions dans leurs prédictions. Dans cette thèse, nous proposons des fonctions de coût sensible à la topologie pour résoudre ces problèmes et améliorer la topologie des reconstructions. Nous commençons par introduire une fonction de coût basée sur la connectivité pour extraire la structure de réseaux à partir d'images 2D. Nous exprimons la connectivité des structures curvilignes en termes de déconnexions qu'elles créent entre les régions de l'arrière-plan de l'image. Notre fonction de coût est conçue pour empêcher ces connexions indésirables entre des régions de l'arrière-plan, et donc de combler les lacunes dans les structures prédites. Ensuite, nous nous concentrons sur l'utilisation de l'homologie persistante (HP) pour améliorer la qualité topologique des reconstructions. Nous proposons une nouvelle technique de filtration en fusionnant deux approches existantes : la filtration par seuillage et la fonction de hauteur. Avec la technique proposée, nous incluons l'information de localisation des erreurs topologiques et augmentons le pouvoir descriptif de l'HP. Afin de faire face aux annotations imprécises, nous proposons une fonction de coût basée sur un modèle de contour actif. Nous traitons les annotations comme des modèles de contour et les laissons se déformer elles-mêmes sur les lignes médianes correctes pendant l'entraînement tout en préservant la topologie de la structure. Notre dernière contribution est d'étendre la fonction de coût basée sur la connectivité mentionnée ci-dessus pour travailler également avec des données 3D. Nous y parvenons en calculant le coût sur des projections 2D. Avec cette méthode, nous réduisons également l'effort d'annotation nécessaire pour fournir des données d'entraînement. Nous démontrons, dans des expériences sur des jeux de données 2D (images satellites de routes et de canaux d'irrigation) et 3D (Angiographie par

Résumé

Résonance Magnétique, Microscopie à Deux Photons), que nos fonctions de perte améliorent significativement la qualité topologique des reconstructions.

Mots-clés : Vision par ordinateur, Apprentissage profond, Délimitation, Structures curvilignes, Connectivité

Contents

Acknowledgements	i
Abstract (English/Français/Deutsch)	iii
List of Figures	5
List of Tables	9
1 Introduction	11
1.1 Existing Methods	11
1.1.1 Automated Delineation	11
1.1.2 Graph-Based Models	13
1.1.3 Embedding Topological Information in Loss Function	13
1.1.4 Persistent Homology	14
1.2 Contribution	14
2 Promoting Connectivity of Network-like Structures by Enforcing Region Separation	17
2.1 Introduction	17
2.2 Related Work	20
2.2.1 Connectivity-oriented loss functions	20
2.2.2 Connectivity-oriented neural architectures	21
2.2.3 Affinity learning	21
2.3 Method	22
2.3.1 Regression Loss: L_{MSE}	24
2.3.2 Connectivity Loss: L_{TOPO}	24
2.4 Experiments	26
2.4.1 Datasets	27
2.4.2 Baselines	27
2.4.3 Network Architecture and Training Details	28
2.4.4 Performance Measures	29
2.4.5 Comparative Results	29
2.4.6 Ablation Study	33
2.5 Conclusion	39
3 Enforcing Connectivity of 3D Linear Structures Using Their 2D Projections	41

Contents

3.1	Introduction	41
3.2	Related Work	42
3.2.1	Delineation of 3D linear Structures	42
3.2.2	Topology-Aware Loss Functions	42
3.3	Method	43
3.3.1	Connectivity loss	43
3.3.2	Projected Connectively Loss	44
3.3.3	Total Loss	45
3.4	Experiments	45
3.4.1	Datasets	45
3.4.2	Metrics	46
3.4.3	Architectures and Baselines.	46
3.4.4	Results	49
3.4.5	Ablation Study	49
3.5	Conclusion	50
4	Persistent Homology with Improved Locality Information for more Effective Delineation	51
4.1	Introduction	51
4.2	Related Work	52
4.2.1	Losses Designed to Enforce Topological Correctness.	53
4.2.2	Losses that rely on Persistent Homology	53
4.3	Method	54
4.3.1	Persistent Homology	54
4.3.2	Training Deep Networks using PH	55
4.3.3	Accounting for the Location of Topological Features during Filtration	56
4.4	Experiments	58
4.4.1	Correlation to the Number of Topological Errors	59
4.4.2	Performance in Training Deep Networks	59
4.4.3	Ablation Study	64
4.5	Conclusion	67
5	Adjusting The Ground Truth Annotations for Connectivity-Based Learning to Delineate	69
5.1	Introduction	69
5.2	Related Work	71
5.2.1	Automated Delineation	71
5.2.2	Handling Noisy Annotations	72
5.2.3	Deformable Contour Models	72
5.3	Method	73
5.3.1	Standard Training Procedure	73
5.3.2	Overview of our Approach	74
5.3.3	Annotations as Network Snakes	75

5.3.4	Computing the Gradients of the Loss Function	75
5.3.5	Speeding Things Up	76
5.4	Experiments	79
5.4.1	Datasets	79
5.4.2	Metrics	79
5.4.3	Architectures and Training Details	80
5.4.4	Label Correction Baselines	83
5.4.5	Comparative Evaluation.	84
5.4.6	Perfectly Accurate Annotations	84
5.4.7	Increasing Annotation Inaccuracy	85
5.4.8	Reducing Annotation Effort	86
5.4.9	Ablation Study	87
5.5	Conclusion	89
6	Conclusion	91
6.1	Summary	91
6.2	Future Work	92
	Bibliography	102
	Curriculum Vitae	103

List of Figures

1.1	Difficulties of annotating data. Most annotation tools take several points on the curvilinear structure as an input from the annotator and then use a tracing algorithm which connects these points while minimising a cost function which depends on the length of the path and voxel intensities along the path. (a) Intensity changes in the curvilinear structures, as it can be seen inside the red circle, renders it almost impossible to determine the centerlines. Furthermore, such changes deteriorates the performance of the tracing algorithms. (b) Annotation tools do not always trace the centerlines. It can create shortcuts which plague the annotations with deviations from the actual centerline.	12
2.1	We enforce road connectivity by penalizing connections between background regions. (a) Input image and ground truth. (b) A distance map predicted by a U-Net trained <i>without</i> our connectivity loss, and its skeletonization, thickened for visibility. Note that, even though there is a gap between road pixels P_1 and P_2 , they remain connected both in the ground truth and in the prediction, because alternative paths exist in the loopy road network. By contrast, background regions A and B connect in the prediction, but not in the ground truth. (c) A distance map predicted by a U-Net trained using our disconnectivity loss and its skeletonization. Our loss function penalizes connections between A and B , preventing gaps in the predicted road.	19
2.2	Handling a potential misalignment between the ground-truth and predicted networks in the image of Fig. 2.1. The lines have been thickened for better visibility. This misalignment makes imposing connectivity constraints on the pixels belonging to the ground-truth centerline impractical. Instead, we only require that background regions far away from the roads and delimited by a dashed line be disconnected.	22

2.3	Computing L_{disc}. We first tile the ground truth annotation and the distance map computed by our network (1). We use the ground-truth roads to segment each tile into separate regions (2). When there are unwarranted gaps in the distance map, there is a least one path connecting disjoint regions such that the minimum distance map value along that path is not particularly small. We therefore take the cost of the path to be that minimum value (3) and we add to our loss function a term that is the maximum such value for all paths connecting points in the two distinct regions (4). This penalizes paths such as the one shown here and therefore promotes the road graph connectivity.	23
2.4	Comparative results on the <i>RoadTracer</i> dataset. For our results, we overlaid the graphs on the inferred distance maps.	32
2.5	Comparative results on the <i>DeepGlobe</i> dataset. For the results of our method, we overlaid graphs on the inferred distance maps.	33
2.6	Comparative results on the <i>Canals</i> dataset. For the results of our method, we overlaid graphs on the inferred distance maps.	34
2.7	Qualitative results on the <i>Massachusetts</i> dataset. For our method, we overlaid graphs on the inferred distance maps.	34
2.8	Effect of changing α in Eq. 2.2 on the distance map the neural network outputs. As α increases, the road map becomes more complete until α becomes so large that it promotes spurious connections even where no roads are present in the image.	35
2.9	Effect of changing β in Eq. 2.4 on the distance map the neural network outputs. As β is increases, the predictions become more precise. It reduces false positive roads until β becomes so large that it creates disconnections on actual roads.	36
2.10	A comparison between the results obtained with the CE and MSE losses, on the <i>RoadTracer</i> data set. For the results, we overlaid graphs on the inferred distance maps.	38
3.1	The intuition behind L_{TOPO} . (a) In a perfect distance map, any path connecting pixels on the opposite sides of an annotation line (dashed, magenta) crosses a zero-valued pixel (red circle). (b) If a distance map has erroneously high-valued pixels along the annotation line, the maximin path (violet) between the same pixels crosses one of them (red circle). (c) The connectivity-oriented loss L_{TOPO} is a sum of the smallest values crossed by maximin paths connecting pixels from different background regions. The background regions are computed by first dilating the annotation (dilated annotation shown in white), to accommodate possible annotation inaccuracy.	44

3.2	Disconnections in 3D linear structures appear in at least two out of three orthogonal projections, unless the structure is occluded.	45
3.3	Median and quartiles over five training runs of scores attained by networks trained with different loss functions. Minimizing our topology-aware loss results in significantly higher score values than minimizing the baselines.	47
3.4	Qualitative comparison of the test results in three different datasets. The connectivity improves significantly when our approach is used.	48
4.1	2D and 3D delineation. (a) Aerial image and slice of a microscopy stack. (b) A network trained using a standard homology-based loss yields road and neurite interruptions. (c) One trained using our localized loss is more topologically accurate and produces predictions that closely resemble the ground truth (d).	53
4.2	Filtration. When the distance map shown on the left is filtered by thresholding, the loop h emerges at scale b^h and is filled at scale d^h . This gives rise to the point (b^h, d^h) in the persistence diagram shown on the right. Here, thresholding means retaining all pixels whose value is lower than the threshold.	54
4.3	Comparing filtration functions on synthetic data. The binary ground truth road annotation (<i>top-left</i> in each table part) contains four loops, marked with cyan dashed lines. We synthesized a predicted class affinity map (<i>bottom-left</i> in each part) by extending one road to the left and interrupting another. In consequence, loop B and D from the ground truth are joined into B' in the prediction, and A is split into A' and E'. For each filtration method, we show binary masks resulting from filtration at different scales, pairs of persistence diagrams, and their optimal matches.	57
4.4	Sensitivity of the topological loss term C to the number of injected errors (a) Ground truth distance maps of road networks. (b) Distance maps corrupted by introducing false roads and interruptions. We randomly injected one error at a time, obtaining corrupt distance maps with 30 errors. We repeated this simulation 10 times. (c,d) The cumulative distribution function of change in the loss term in response to injecting one error. In (c), C is evaluated using the filtration by thresholding distance maps, whereas in (d) we use our filtration. The probability of decreasing the existing loss term by injecting additional errors is around 0.4, whereas for our loss term it drops to 0.2. We conclude that our loss term is more monotonic with respect to the error number.	59
4.5	Qualitative results on the <i>Massachusetts</i> dataset.	64
4.6	Comparative results on the <i>RTracer</i> dataset.	65
4.7	Comparative results on the 3D <i>Neurons</i> dataset.	66

List of Figures

- 5.1 **Our approach.** To account for annotation inaccuracies during training, we jointly train the network and adjust the annotations while preserving their topology. 70
- 5.2 **Correcting an inaccurate annotation.** (a) A microscopy scan of a neurite with an inaccurate annotation overlaid in white. (b) Distance map predicted by the deep net. Ideally, the pixels crossed by the centerline should have value zero (dark color). In practice, this is not always the case. There are non-zero values in the area indicated by the red arrow, presumably because the neurite is hardly visible there. Nevertheless, the distance map is sufficiently good to adjust the annotation. The adjusted annotation is shown in (c) and (d). This network retrained with adjusted annotations can now generate a better distance map even where the neurite is barely visible. 70
- 5.3 The three approaches to training described in Sec. 5.3.4 and 5.3.5. In *SnakeFull*, the training objective is also used as the objective of the snake. This makes some gradient components vanish, simplifying gradient computation, but results in snake updates that are costly to compute. *SnakeFast* can accommodate an arbitrary snake objective, which makes it faster than *SnakeFull*, even though it requires backpropagation through a sequence of snake updates. In *SnakeSimple*, the backpropagation over the snake updates is simply omitted. This approach is the fastest. We analyze the accuracy vs. speed tradeoff induced by these three methods in section 5.4. 77
- 5.4 **Compared behavior of *SnakeSimple*, *SnakeFast*, and *SnakeFull* on a synthetic 2D example.** (*Left column*) At the bottom, distance map and corresponding annotation. At the top, we simulated an unwarranted break in the distance map (horizontal yellow line) and shifted the annotation by several pixels. (*Other Columns*) In three separate runs, we performed 100 Gradient Descent using either *SnakeFull*, *SnakeFast*, or *SnakeSimple*. In the *top row*, we show the corrected annotation and the updated distance maps. The *bottom row* depicts the differences between the updated maps and the ground-truth one. We also indicate the computation times. *SnakeFull* removes the interruption in the distance map but the computation is slow. *SnakeFast* is much faster and fills the gap in the distance map almost as well. *SnakeSimple* is even faster but yields a corrected annotation that is too short, as highlighted by the red arrow. 78
- 5.5 Test predictions of different methods on three data sets. The green ellipses denote areas where training with the original annotations results in unwarranted breaks in the delineations whereas our approach does not. 82
- 5.6 Qualitative comparison of the results of *SnakeFast* to existing methods of training with noisy labels. The green ellipses denote areas where baselines result in unwarranted breaks in the delineations at test time whereas our approach does not. 83
- 5.7 Results of training with the precise annotations of the *Synthetic* data set. When the annotations are precise, *SnakeFast* performs as well as training with the *OrigAnnot*. 85

5.8	Annotation Deformation Levels. The deformation magnitude increases from left to right.	85
5.9	Increasing the amount of deformation. <i>APLS</i> and <i>TLTS</i> scores as a function of the deformation level. The <i>OrigAnnot</i> scores decrease fast whereas those of <i>SnakeFast</i> decrease much more slowly.	86
5.10	Coarse Annotations (a) Training image of a neurite (b) Distance map obtained from original annotation overlaid in red (c) Distance map obtained from coarse annotation overlaid in red. Coarse annotations are obtained by connecting neurite end points and bifurcations with straight lines, and are easier to perform than full annotations.	87
5.11	Results of training a <i>UNet</i> with <i>OrigAnnot</i> and <i>SnakeFast</i> on the <i>Neurons</i> data set with easy annotations.	87

List of Tables

2.1	Comparative results on the <i>RoadTracer</i> dataset [6]. Our loss function makes even the simple <i>UNet</i> attain state of the art performance. Computing the loss in windows results in improvement of four out of five performance criteria. The results for our method are means and standard deviations over three independent training runs.	30
2.2	Comparative results on the <i>DeepGlobe</i> dataset. Our loss function improves the performance of both <i>UNet</i> and <i>DRU</i> in terms of all the metrics, with <i>DRU</i> attaining the state-of-the-art performance. The results for our method are again means and variances over three independent training runs.	30
2.3	Comparative results on the <i>Canals</i> dataset. Our loss function boosts the performance of both <i>UNet</i> and <i>DRU</i> in terms of all the five metrics. The results for our method are means and standard deviations over three independent training runs.	31
2.4	Comparative results on the on the <i>Massachusetts</i> dataset. Using our loss function combined with <i>UNet</i> outperforms [48] on all metrics. For a fair comparison, we used the same <i>UNet</i> architecture as in [48].	31
2.5	Impact of changing the value <i>alpha</i> that balances L_{MSE} against L_{TOPO} in Eq. 2.2, using <i>UNet+TOPO-win</i> on the <i>RoadTracer</i> dataset. The window size is fixed to 64x64 and β to 0.1. The corresponding results are depicted qualitatively by Fig. 2.8.	35
2.6	Impact of changing the value of β that balances the connectivity and dis-connectivity components of our loss in Eq. 2.4, using <i>UNet+TOPO-win</i> on the <i>RoadTracer</i> dataset. The corresponding results are depicted qualitatively by Fig. 2.9. The bottom part of the table features the performance measures obtained when <i>not</i> using the L_{MSE} loss. They are lower, which shows that balancing the connectivity and dis-connectivity components of L_{TOPO} is not enough to ensure maximum performance.	36

List of Tables

2.7	The impact of window size on performance. Results of experiments on the <i>RoadTracer</i> dataset. α is fixed to $1e-3$ and β to 0.1. <i>UNet+TOPO-win</i> is used in all experiments.	37
2.8	The impact of changing the window size on performance. Results of experiments on the <i>DeepGlobe</i> dataset. α is fixed to $1e-3$ and β to 0.1. <i>UNet+TOPO-win</i> is used in all experiments.	37
2.9	Comparison of Cross Entropy and Mean Square Error. Results of experiments on the <i>RoadTracer</i> dataset [6].	38
3.1	Comparative results. A <i>UNet</i> trained with our loss function outperforms existing methods by a considerable margin in terms of the topology-aware metrics. The improvement in terms of the pixel-wise metrics is smaller but still there on average.	46
3.2	We varied α to investigate its impact on the test performance in <i>Neurons</i> dataset. The results show that setting this coefficient too-low or too-high perturbed performance, and its optimal value is in the order of $1e-3$	49
3.3	We varied β to investigate its impact on the test performance in <i>Neurons</i> dataset. According to all the performance measures, the best results are obtained for $\beta = 0.1$	50
4.1	Validation results on the <i>Massachusetts</i> dataset. Our loss function outperforms all PH-based loss functions. We report means and standard deviations over three independent training runs.	62
4.2	Our loss function outperforms all PH-based loss functions on the <i>RTracer</i> dataset. We report means and standard deviations over cities from the test set.	63
4.3	Comparative results on the <i>Neurons</i> dataset. Our loss outperforms all the baselines. We report means and standard deviations over three independent training runs.	63
4.4	Comparative results on the <i>Brain</i> dataset. Our loss outperforms all PH-based losses. Means and standard deviations over three independent training runs as presented.	63
4.5	Impact of changing the learning coefficient of localized PH loss on the <i>Massachusetts</i> dataset. The window size is fixed to 64x64.	66
4.6	Impact of changing the window size when computing our localized loss on the <i>Massachusetts</i> dataset. The learning coefficient is fixed to $1e-2$	67

4.7	Performances of different height functions used for localized PH loss on the <i>Massachusetts</i> dataset. The learning coefficient is fixed to $1e-2$ and window size to 64×64	67
5.1	Performance of deep nets trained with different loss functions on our three data sets and the time needed for single training iteration.	81
5.2	Performance of a <i>UNet</i> trained with the <i>OrigAnnot</i> and with <i>SnakeFast</i> on the <i>Synthetic</i> data set with precise annotations.	84
5.3	Performance of <i>UNet</i> trained using <i>SnakeFast</i> and <i>OrigAnnot</i> on the <i>Neurons</i> data set with very coarse annotations. Performance of <i>UNet</i> trained using the precise annotations shown for reference.	86
5.4	Performance of <i>UNet</i> trained using <i>SnakeFast</i> on the <i>Neurons</i> data set when varying the elasticity and spring term coefficients.	88
5.5	Performance of <i>UNet</i> trained using <i>SnakeFast</i> on the <i>Neurons</i> data set when varying the inverse stepsize, together with the number of snake updates used in every training iteration and the resulting iteration time.	88
5.6	Performance of deep nets trained with <i>MAE</i> and <i>MSE</i> costs on the <i>Neurons</i> data set and the time needed for single training iteration.	89

1 Introduction

Curvilinear structures are ubiquitous in nature, appearing on a wide range of scales, from microscale to macroscale. They are observable in numerous contexts, including biomedical images, satellite images, geological images, and others. The reconstruction of neural circuits in the brain, the detection of fractures in materials, and the extraction of road and irrigation canal networks from satellite images all rely heavily on these structures.

Reconstruction of curvilinear structures is a critical task in many different domains. Neurons are the fundamental building blocks of the nervous system, and their morphology plays a crucial role in understanding the underlying neural circuitry. Delineation of neurons in microscopy images is essential for studying their morphology, connectivity, and function. Road and canal networks are essential for transportation planning, urban development, and disaster management. Delineation of road networks in satellite data is necessary for automatic road extraction, updating of road maps, and real-time traffic monitoring. However, manual extraction of these structures can be very expensive and time consuming. Furthermore, especially in 3D data, annotation tools for manual extraction, are not always easy to use or accurate. Automated systems for delineation is essential for better and quicker analysis of curvilinear structures. Hence, this task has been a longstanding challenge in Computer Vision field.

1.1 Existing Methods

In this section, we summarise existing delineation methods, and topology-aware approaches for this task.

1.1.1 Automated Delineation

In the early years, deformable models [37, 15], manually designing filters that respond strongly to tubular structures [36, 57], feeding hand-designed features into boosted trees [12], support vector machines [50], and GradientBoost [95] were used for automated delineation. With the

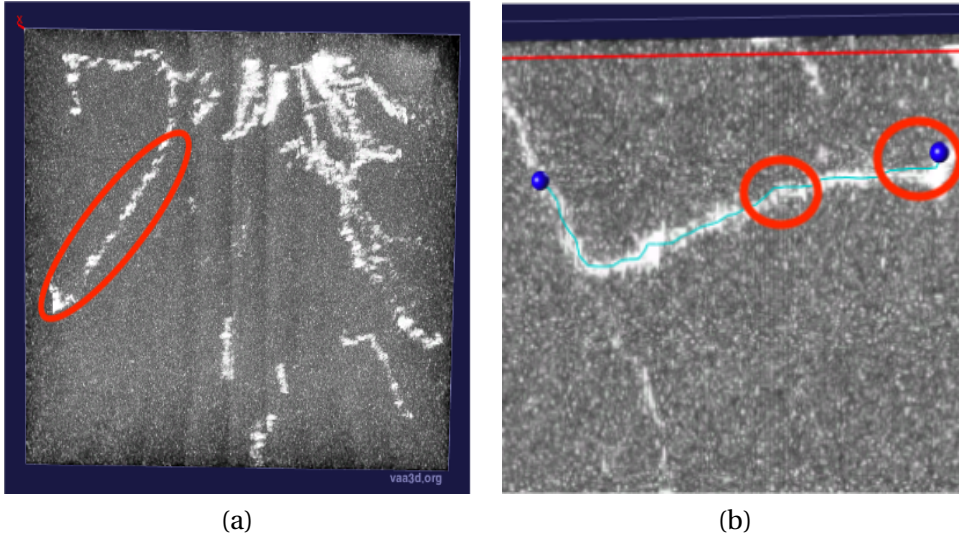


Figure 1.1: **Difficulties of annotating data.** Most annotation tools take several points on the curvilinear structure as an input from the annotator and then use a tracing algorithm which connects these points while minimising a cost function which depends on the length of the path and voxel intensities along the path. (a) Intensity changes in the curvilinear structures, as it can be seen inside the red circle, renders it almost impossible to determine the centerlines. Furthermore, such changes deteriorates the performance of the tracing algorithms. (b) Annotation tools do not always trace the centerlines. It can create shortcuts which plague the annotations with deviations from the actual centerline.

advancement of neural networks, convolutional networks replaced these methods, and most of the recently used models rely solely on deep networks [25, 71, 7, 103, 48]. When properly trained, deep neural networks perform at their highest level. However, they require a large amount of data to be trained properly. Finding reliable annotated data, particularly in 3D, is difficult. It is seldom offered in adequate amounts in practice. Furthermore, the annotated data that does exist is rarely precise since it is difficult to manually annotate curvilinear structures as illustrated in Fig. 1.1. Unfortunately, performance of deep networks depends significantly on the precision of the annotations. The fact that most deep networks are trained by optimizing cross entropy or differentiable intersection-over-union loss can be largely attributed to this. Both are sensitive to even little centerline shifts of the linear structures since they are pixel-wise measurements. This is somewhat addressed in [71] by adding a loss component that takes into account the network output's global statistics, but the cross entropy still plays a significant role in the total loss. Similarly in [49], the approach focuses on adding a topology-preserving term but still relies on precise annotation.

Another main challenge in curvilinear structure delineation is to preserve the topology of the structure during the delineation process. Curvilinear structures often form complex networks with branches, loops, and intersections. Preserving the topology of such structures is crucial for downstream applications. However, most of the current practices cannot capture the topology of these structures successfully. The majority of the currently used methods

rely on neural networks, such as UNet [81], to extract binary masks from images indicating which pixels/voxels are a part of the curvilinear structure and which are not. They regrettably cannot ensure that the connection of the created masks matches the connectivity of the actual network-like structures. This is thus because these models are taught to minimise losses that do not explicitly require topological consistency, such as cross-entropy and mean square error. Networks trained with per-pixel losses produce binary masks plagued with topological errors like disconnections, missed junctions, and false positive connections when the annotations do not exactly match the imaged structures, which is always the case with manual image annotations.

1.1.2 Graph-Based Models

Insufficiency of standard models to capture topology, can be addressed by combining a convolutional encoder with a decoder that instead of outputting a pixel-wise binary mask, displays network-like structures as a graph [6, 62, 26]. The graph is expanded repeatedly throughout inference time, with each step including the neural network adding a new node to the graph while taking current graph state and image data into consideration. These graph-based approaches make it simple to avoid overly punishing extracted curvilinear structures that marginally deviate from their ground truth, as opposed to the approach based on expressing a road map as a binary mask, and to account for existing connection while building the graph. Training these models is more challenging and unstable than training convolutional networks due to the non-differentiability of the node insertion procedure. Furthermore, since the inserted nodes are conditioned on the previous states of the graph, heuristic methods must be used in case the reconstructed graph is deviated significantly from the ground truth.

1.1.3 Embedding Topological Information in Loss Function

In this thesis, we are aiming to introduce loss functions that can teach the topology of the curvilinear structures to standard delineation models. Topology-aware loss functions aim to preserve the topology of curvilinear structures during the delineation process by ensuring that the resulting reconstruction is a valid graph with correct topology. Topology-aware loss functions go beyond pixel-wise classification accuracy by encoding topological information of the structures in the loss function itself. The main challenge with such loss functions is to capture topological differences with a differentiable function.

A number of topology-aware losses have been proposed in the recent literature. Li et al. [61] propose to use connectivity information between neighboring pixel-pairs. With this additional supervision, local connectivity can be enforced but it is still prone to noise in the annotations. In [71], a pre-trained VGG [84] is used to compare the shape features of the predicted probability map and the ground truth. This comparison is then used as additional guidance for the delineation model. However, this approach is heavily based on the assumption that the pre-trained model can highlight topological differences between the prediction and the

ground truth. In reality, there is no guarantee that it can highlight all of them. Other loss functions which explicitly evaluate the topology have been proposed for biomedical and satellite road data. A connectivity-oriented loss function [13, 39] optimizes Rand index of a segmentation map in order to prevent topological errors, but it cannot capture disconnections if there are loops in the data. Another way of comparing topologies is to introduce a differentiable skeletonization algorithm [83] instead of the standard non-differentiable method. This allows the loss function to be computed on graphs instead of probability maps, hence, the topologies of prediction and ground truth can be compared effectively.

1.1.4 Persistent Homology

Persistent Homology (PH) [32, 33, 108] is an well-established topological descriptor which can be used for describing and comparing topologies. By using a filtration function with a tunable parameter, PH extracts connected components (0-homology class), loops (1-homology class) and voids (2-homology class) from the data. Each topological structure is born and dies at a particular value of the tunable parameter. The topology of the object is then represented by persistence diagram which is a diagram of birth and death times of all the homology classes found in the object. Extracted persistence diagrams can be used to find topological similarities across two objects. This approach has been effectively used to train deep networks for a variety of tasks such as delineation [48], image segmentation [48, 28, 27] and crowd counting [1].

Recently, it has been shown that persistence diagrams can be computed for grayscale images by using thresholding as a filtration function [48, 28, 40, 60, 19]. They can be incorporated into a loss function to train a deep network, since they are differentiable with respect to pixel/voxel values. In [27], a loss is used to enforce the desired the Betti number, number of homology classes appearing in the persistence diagram, on the prediction. This approach has been further extended to a loss function that maximise the similarity of prediction and ground truth persistence diagrams computed on binary maps [48]. However, none of these approaches use the full potential of PH because they cannot use the location information of topological errors effectively.

1.2 Contribution

In this thesis, we are introducing topology-aware loss functions that provides reconstructions of network-like structures in 2D and 3D data with better topologies than current practices.

- In chapter 2, we introduce a 2D connectivity-oriented loss which expresses connectivity of curvilinear structures in terms of disconnection of two background regions separated by these structures in the groundtruth.
- In chapter 3, a topology-aware loss function based on Persistent Homology is proposed. We are improving the effectiveness of PH on training deep networks by including the

location information with a new filtration technique.

- In chapter 4, we introduce a loss function that accounts for annotation inaccuracies. We tackle this problem by treating the annotations as active contour models and allowing them to deform over correct centerlines while preserving the topology.
- In chapter 5, we extend the connectivity-oriented loss function introduced in chapter 2, originally limited to two-dimensional images, to also work with three-dimensional scans. We attain this extension by using the loss on two-dimensional projections of the three-dimensional scans.

2 Promoting Connectivity of Network-like Structures by Enforcing Region Separation

In this chapter, we propose a novel, connectivity-oriented loss function for training deep convolutional networks to reconstruct network-like structures, like roads and irrigation canals, from aerial images. The main idea behind our loss is to express the connectivity of roads, or canals, in terms of disconnections that they create between background regions of the image. In simple terms, a gap in the predicted road causes two background regions, that lie on the opposite sides of a ground truth road, to touch in prediction. Our loss function is designed to prevent such unwanted connections between background regions, and therefore close the gaps in predicted roads. It also prevents predicting false positive roads and canals by penalizing unwarranted disconnections of background regions. In order to capture even short, dead-ending road segments, we evaluate the loss in small image crops. We show, in experiments on two standard road benchmarks and a new data set of irrigation canals, that convnets trained with our loss function recover road connectivity so well that it suffices to skeletonize their output to produce state of the art maps. A distinct advantage of our approach is that the loss can be plugged in to any existing training setup without further modifications. This work appeared in [75].

Oner, D., Kozinski, M., Citraro, L., Dadap, N. C., Konings, A. G., and Fua, P. *Promoting Connectivity of Network-Like Structures by Enforcing Region Separation*. IEEE Transactions on Pattern Analysis and Machine Intelligence 2021.

2.1 Introduction

Reconstruction of road networks from aerial images is a classic computer vision problem [3, 93, 79, 35], which remains actively studied to this day [25, 66, 62, 6, 7, 26, 70, 103]. By contrast, the reconstruction of drainage canals has so far remained out of focus of most of the vision community. However, it is of practical importance for hydrologic analysis [72, 59], which is

Chapter 2. Promoting Connectivity of Network-like Structures by Enforcing Region Separation

becoming even more crucial at the time of rapid climate change. Due to their network-like structure, canals are amenable to reconstruction by the same algorithms as roads, and we address these two problems jointly. Most of the existing approaches [25, 71, 7, 103] rely on convolutional networks to extract from images binary masks denoting which pixels belong to roads and which do not. Unfortunately, they do not guarantee that the connectivity of the produced masks corresponds to that of the real road network. This is because these methods are trained to minimize losses, such as cross-entropy and mean squared error, that do not explicitly enforce topological consistency. When the annotations do not perfectly coincide with the imaged structures, which is always the case of satellite image annotations, networks trained with the per-pixel losses produce binary masks plagued by topological errors, such as road interruptions, missed junctions, and false positive connections.

In recent literature, this problem has been addressed by combining a convolutional encoder with a decoder that represents a network of roads as a graph, as opposed to a binary mask [6, 62, 26]. At inference time, the graph is grown iteratively: At each step, the neural network adds a new node to the graph by taking image features and the current state of the graph into account. By contrast to the approach based on representing a road map as a binary mask, these graph-based methods make it easy to prevent excessively penalizing predicted roads that deviate slightly from their ground truth models, and to account for existing connectivity when growing the graph. However, the non-differentiability of the node insertion operation makes training these networks more difficult and brittle than training convnets.

In this chapter, we show that connectivity of road and drainage canal networks can be enforced directly on a convolutional neural net, in a fully differentiable manner, and without the need to represent the graph explicitly. This allows end-to-end training and results in increased performance. Our approach involves relaxing the usual requirement of coincidence of annotated and predicted foreground pixels. Instead, we require that predictions contain uninterrupted sequences of foreground pixels that can deviate by a few pixels from the ground-truth annotations. This enforces connectivity while dealing with possibly imprecise annotations.

The difficulty is to express this requirement in the form of a differentiable loss function that can be used to train a deep network. The central idea of our approach is to forgo enforcing connectivity of the pixels annotated as centers of roads or canals, which may not coincide with true roads or canals. Instead, we express the connectivity of the annotated structures in terms of the disconnections that they create between regions annotated as background. More precisely, we require that two regions separated by a line in the ground truth, are also separated in the prediction. As shown in Fig. 2.1, this effectively enforces continuity of the predicted road or canal networks. By requiring that connected components of pixels annotated as background remain connected in the prediction, we prevent predicting false positive road or canal segments. In other words, we re-purpose the differentiable machinery proposed in the MALIS segmentation algorithm [13, 39] to enforce the dis-connectivity of image regions separated by a road. To capture dead-ending segments, we compute our loss in small image

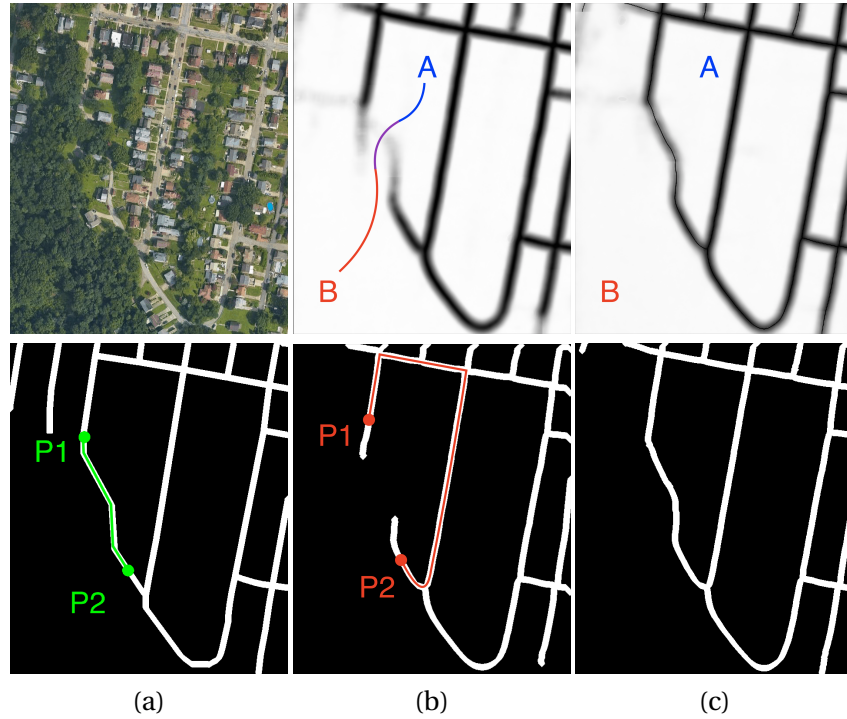


Figure 2.1: **We enforce road connectivity by penalizing connections between background regions.** (a) Input image and ground truth. (b) A distance map predicted by a U-Net trained *without* our connectivity loss, and its skeletonization, thickened for visibility. Note that, even though there is a gap between road pixels P_1 and P_2 , they remain connected both in the ground truth and in the prediction, because alternative paths exist in the loopy road network. By contrast, background regions A and B connect in the prediction, but not in the ground truth. (c) A distance map predicted by a U-Net trained using our disconnection loss and its skeletonization. Our loss function penalizes connections between A and B , preventing gaps in the predicted road.

windows, which are likely to be subdivided even by short road and canal sections.

Our contribution therefore is a novel approach to enforcing global connectivity of reconstructions of network-like structures from images. It can be used to boost the performance of *any* road delineation deep network, without having to change the network itself. This is in stark contrast to the graph networks that do require changing both the network architecture and the training procedure. We demonstrate on both roads and drainage canals, that a simple U-Net [81] trained with our loss function, and combined with a standard skeletonization algorithm, attains state of the art performance in terms of the connectivity of the reconstructed networks.

2.2 Related Work

The existing approaches to reconstruction of networks of drainage canals rely on dedicated sensing modalities, like multi-spectral imaging and lidar [94, 51], and require extensive user interaction. We show that the canals can be reconstructed from visual spectrum satellite images and with little required correction, just like roads.

Many existing road segmentation algorithms rely on convnets [25, 7, 71, 103, 70] and all of them face the same difficulty: Training them by minimizing a cross-entropy loss, which is a *local*, pixel-wise measure, does not guarantee that their output preserves the global connectivity of road networks. Training the network to multi-task and to find not only the road centerline but also its spatial extent [25, 7, 61], its orientation [7, 61], or edges [61], mitigates the problem but does not explicitly enforce better connectivity. We instead propose to explicitly define the loss function to evaluate the connectivity.

2.2.1 Connectivity-oriented loss functions

Ours is not the first attempt to make a convnet capture connectivity of linear structures in images by incorporating connectivity-oriented terms in the loss function. One existing approach [61] is to use the connectivity between neighboring pixel pairs as an additional source of supervision. This enforces the local connectivity of the ground truth on the prediction, but does not handle annotation misalignment. By contrast, our method focuses on long-range connectivity of roads with possibly misaligned annotations. Another approach [71] is to introduce a perceptual loss function that depends on the statistical differences between features computed by forwarding either the ground truth or the prediction through a pre-trained neural network. While this loss is indeed non-local, and has been shown to improve the connectivity of the predictions, it does not model connectivity explicitly. Instead, it heavily relies on the assumption, that a pre-trained neural network implicitly captures some topological properties of the input. By contrast, our loss function models connectivity explicitly.

Loss functions explicitly evaluating the topology of the predicted masks have been proposed for medical image segmentation [28, 48]. However, strictly topological techniques are focused on counting loops and connected components in the data, irrespectively of their spatial position, and cannot distinguish between different branching patterns. That makes topological methods a good choice when the segmented object has a relatively simple topology, like the aortic valve, but not well suited for roads, which exhibit complex branching patterns and form numerous loops. In [48], this limitation is overcome by evaluating the topological loss only in small image patches, which generalizes this method to more complex topologies, including roads. By contrast, our topological loss can handle complex loopy structures even without the patch-wise approach. However, we will see that using it boosts its performance especially when there are dead-ending roads and increases its numerical stability.

2.2.2 Connectivity-oriented neural architectures

Problems with connectivity can be addressed by designing predictors that output graphs instead of per-pixel masks, and explicitly decide about the presence of connections between map nodes. This can be done as a post-processing step by generating a pool of potential additional connections and training a classifier to decide which of the candidates should be inserted into the network [66, 70]. A drawback of this approach is that it is not end-to-end trainable.

A more elegant alternative is to use graph neural networks to predict the road graphs directly from the images [6, 62, 26]. This approach has certain disadvantages. Inference consists in a sequence of non-differentiable node insertion operations, which makes such networks slower than convnets. They are also more difficult to train, because node insertion is conditioned on the current state of the graph, and heuristics are needed to decide what is the optimal operation when the graph built so far is inconsistent with the ground truth. In our experimental evaluation, we show that a simple convnet can outperform these approaches when trained with our loss function and post-processed with a vanilla skeletonization. However, we still think that predicting graphs from images has merit, and the idea could be applied on top of a convnet trained with our loss.

2.2.3 Affinity learning

To enforce region connectivity, we use the maximin formulation of MALIS [13, 39], a connectivity-oriented approach to segmenting cells in electron microscopy images of neural tissue. It relies on the observation, that the predicted strength of connection between a pair of pixels can be expressed as the lowest value that needs to be crossed when traveling between the pixels in the prediction. If this value equals θ , thresholding the prediction with $\theta' < \theta$ produces a connected component containing both pixels. Thresholding the prediction with $\theta'' > \theta$ breaks the connection between the pixels. Formally, θ is called a maximin cost of a pixel pair, and MALIS incorporates it into a differentiable loss term which is maximized for all pairs of pixels that belong to the same annotated cell, and minimized for all pairs of pixels from different cells. Recently, MALIS has been further perfected by constraining the maximin cost of a pair of end points belonging to the same cell to originate from a pixel located at the same cell as the end points [39], which stabilizes training at its early iterations.

We could have used MALIS to enforce the connectivity of road or canal pixels in the output of a segmentation network. However, we will see in the results section that it is less effective. This is for two reasons. First, both roads and canals often form loops and even if a connection between two road pixels is missed, they may still be connected via a different path. As illustrated by Fig. 2.1, there is a gap between pixels P_1 and P_2 . Yet they are still connected to each other. Hence, this disconnection cannot be fixed simply by enforcing connectivity of any road pixel pairs. Second, road and canal annotations usually take the form of one-pixel-thick centerline delineations that are rarely precise. Strictly enforcing the connectivity of pixels annotated as

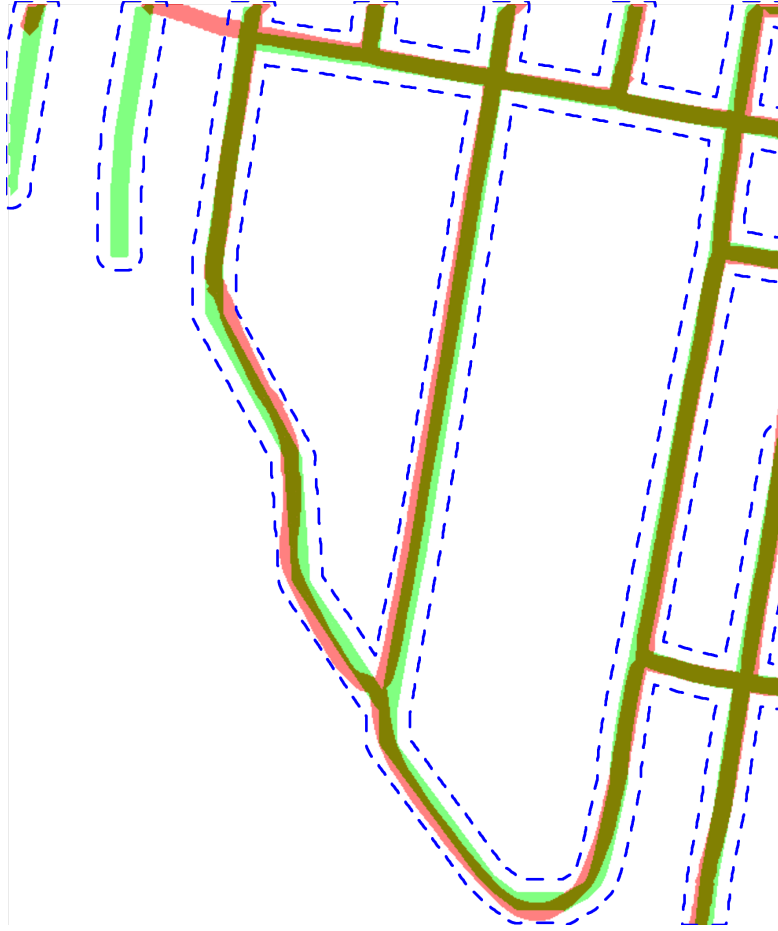


Figure 2.2: Handling a potential misalignment between the **ground-truth** and **predicted networks** in the image of Fig. 2.1. The lines have been thickened for better visibility. This misalignment makes imposing connectivity constraints on the pixels belonging to the ground-truth centerline impractical. Instead, we only require that background regions far away from the roads and delimited by a **dashed line** be disconnected.

roads, but not overlapping with true roads, would confuse the network and negatively impact its precision. In particular, topologically correct predictions could still be penalized, when misaligned with the ground truth, as illustrated by Fig. 2.2. We therefore use MALIS to enforce the disconnections between background regions, separated by road annotations, and at a distance of at least 10 pixel from them. The dashed blue lines in Fig. 2.2 materialize their boundaries.

2.3 Method

Given a training set of N aerial images $\{x_i\}_{1 \leq i \leq N}$ and corresponding ground-truth binary masks $\{y_i\}_{1 \leq i \leq N}$ representing the roads or drainage canals in these images, we want to train a deep network $f_{\Theta}(\cdot)$, with weights Θ , that takes an image x as input and returns a distance

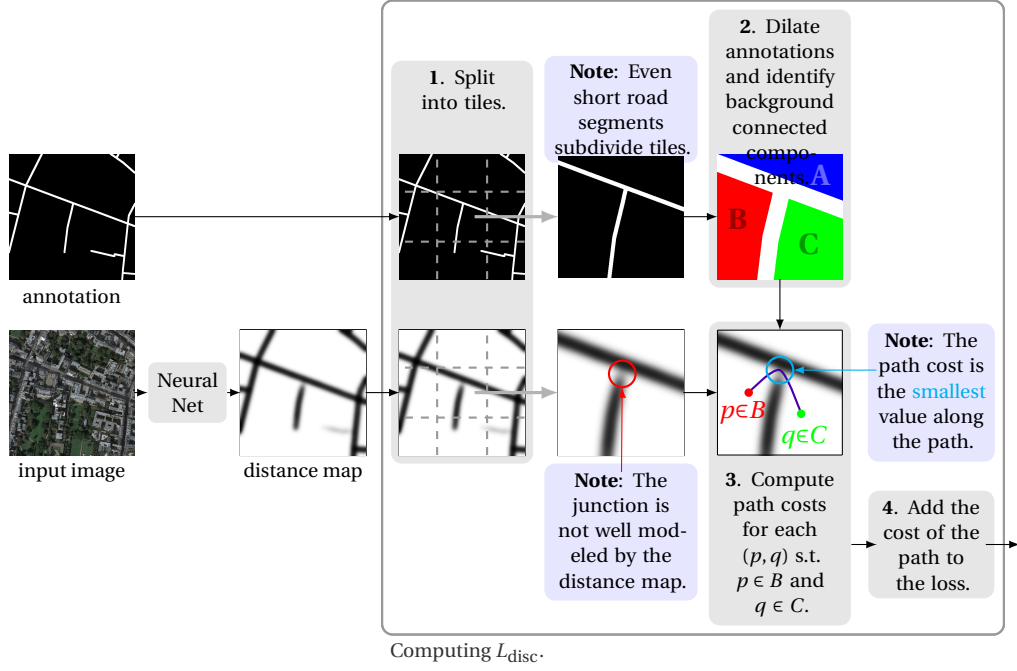


Figure 2.3: **Computing L_{disc} .** We first tile the ground truth annotation and the distance map computed by our network (1). We use the ground-truth roads to segment each tile into separate regions (2). When there are unwarranted gaps in the distance map, there is a least one path connecting disjoint regions such that the minimum distance map value along that path is not particularly small. We therefore take the cost of the path to be that minimum value (3) and we add to our loss function a term that is the maximum such value for all paths connecting points in the two distinct regions (4). This penalizes paths such as the one shown here and therefore promotes the road graph connectivity.

map \hat{y} , consistent with the ground-truth. Our goal is to ensure that \hat{y} represents the same connectivity as y . To this end, we minimize

$$R(\Theta) = \sum_i L(y_i, f_{\Theta}(x_i)), \quad (2.1)$$

$$L(y, \hat{y}) = L_{\text{MSE}}(y, \hat{y}) + \alpha L_{\text{TOPO}}(y, \hat{y}), \quad (2.2)$$

with respect to the network weights Θ . Here the loss function L is the sum of two terms L_{MSE} and L_{TOPO} , and α is a parameter of the method that we set empirically using a validation set. L_{MSE} is a regression loss, used to train the network to predict the distance from each pixel to the center of the closest road or canal as in [85]. This lets us penalize the deviation of the predicted road center from its annotated position more gently than when using the more standard cross entropy. Allowing for these deviations enables the connectivity-oriented L_{TOPO} to force the network to predict uninterrupted roads and canals even if they do not coincide perfectly with the annotations. We describe both terms below in more detail.

Chapter 2. Promoting Connectivity of Network-like Structures by Enforcing Region Separation

2.3.1 Regression Loss: L_{MSE}

We define L_{MSE} as the Euclidean norm of the difference between the predicted distance map \hat{y} and the distance map y_D generated from the ground truth binary mask y

$$L_{MSE}(y, \hat{y}) = \sum_{p \in I} (\hat{y}[p] - \min(y_D[p], D_{\max}))^2, \quad (2.3)$$

where $X[p]$ denotes the value of image X at pixel p , I is the set of pixel indices in the input image. The magnitude of the ground truth distance map for pixels that are far away from foreground structures can be large, and we found it advantageous to cap the ground truth distance at $D_{\max} = 20$ pixels, found empirically.

L_{MSE} can be used by itself to train the network. As shown in Fig. 2.1, this gives good results in terms of per-pixel precision but the resulting binary masks feature many unwarranted interruptions. To prevent this, we now turn to the second term of Eq. 2.2.

2.3.2 Connectivity Loss: L_{TOPO}

L_{TOPO} is the topology term that we adapt from [39] to penalize, in a differentiable manner, unwanted interruptions and false connections in the output distance maps. Since, as explained in section 2.2.3, directly penalizing the interruptions and false connections of the foreground is ineffective in our task, we formulate our loss function in terms of connectivity of the background regions. This indirect approach to enforcing connectivity is the main contribution. As shown in Fig. 2.1, an erroneous break in a predicted road causes two background regions, separated by a road in the ground truth mask y , to touch in the distance map \hat{y} produced by the network. The first component of our loss, L_{disc} , penalizes such contacts. Similarly, a false positive road divides a small crop of the predicted distance map into two background regions, while the same crop of the ground truth distance map contains a single connected component of the background. Such errors are penalized by the second component of our loss, L_{conn} . The full topological loss takes the form

$$L_{TOPO}(y, f(x)) = L_{disc}(y, f(x)) + \beta L_{conn}(y, f(x)), \quad (2.4)$$

where β is a parameter of the loss that balances the effects of these two terms. We introduce L_{disc} and L_{conn} below.

Maximin Dis-Connectivity.

As illustrated in Fig. 2.1, in order to discourage interruptions of a predicted road, we identify all pairs of background regions that the road separates in the ground truth, and penalize connections between these regions in the predicted distance map. To that end, we follow the *maximin* approach of Turaga et al. [13]. Intuitively, since the value of a road or canal pixel in a correct distance map should be small, and the background pixels should be large,

two background pixels in an image can be considered connected if there exists a path of large-valued pixels between them. The ‘strength’ of this connection, can be evaluated as the value of the smallest pixel on the path with the largest smallest pixel of all paths connecting the end points. This path is depicted as the violet line in box 3 in Fig. 2.3, and its smallest pixel is encircled in blue. Therefore, for each pair of pixels that are separated by a road or canal in the ground truth, we make L_{disc} express the ‘strength’ of the connection between them. As a result, minimizing L_{disc} ensures the dis-connectivity of regions on the opposite sides of roads and canals and, indirectly, improves the connectivity of roads and canals.

The detailed computation of L_{disc} is depicted in Fig. 2.3. We first dilate the centerline annotations by 5 pixels, which corresponds to the largest displacement between the image and the annotation that we have observed in our training data. We can therefore assume all the road pixels belong to this dilated region, which we denote as \mathcal{R} and which can also contain non-road pixels. Let \mathcal{B} be the set of background regions, that is, connected components in the remainder of the image. Let us consider two pixels $q \in A$ and $r \in B$ such that $A, B \in \mathcal{B}$ and $A \neq B$. Intuitively, q and r lie on different sides of an annotated road. Since road pixels should receive low predictions, a path π connecting q and r crosses a predicted road in the distance map \hat{y} if, for at least one point p along the path, $\hat{y}[p]$ is close to zero. We therefore define the cost of path π in the predicted distance map \hat{y} as $c(\pi, \hat{y}) = \min_{p \in \pi} \hat{y}[p]$, and measure the ‘connectivity’ between background pixels q and r , in terms of the maximin cost $d_{\text{maximin}}(\hat{y}, q, r) = \max_{\pi \in \Pi(q, r)} c(\pi, \hat{y})$, where $\Pi(q, r)$ is the set of all paths connecting q and r . We enforce road connectivity by minimizing the maximin cost for all pairs of pixels that are separated by a road in the ground truth. To that end, we define our connectivity-enforcing loss as

$$L_{\text{disc}}(y, \hat{y}) = \sum_{A, B \in \mathcal{B}, A \neq B} \sum_{q \in A, r \in B} d_{\text{maximin}}(\hat{y}, q, r)^2. \quad (2.5)$$

When computed naively, the loss, Eq. 2.5, requires summing costs over pairs of pixels, which would be computationally expensive. However, Turaga et al. [13, 39] have shown that, because $d_{\text{maximin}}(\hat{y}, q, r)$ is equal to the value of the smallest pixel that has to be visited when traveling between q and r in the prediction \hat{y} , L_{disc} can be computed efficiently as a sum over pixels, as opposed to pixel pairs, as

$$L_{\text{disc}}(y, \hat{y}) = \sum_{p \in \mathcal{R}} w_p \hat{y}[p]^2, \quad (2.6)$$

where w_p counts the pairs of pixels whose maximin cost is equal to $\hat{y}[p]$. Formally, we denote the maximin path between a pair of pixels q, r by $\pi(q, r)$ and define

$$w_p = \sum_{A, B \in \mathcal{B}, A \neq B} \sum_{q \in A, r \in B} \mathbb{1}[p = \arg \min_{\rho \in \pi(q, r)} \hat{y}[\rho]], \quad (2.7)$$

where $\mathbb{1}[\cdot]$ is the indicator function. The algorithm for computing the w_p ’s is based on the Kruskal’s maximum spanning tree algorithm, and we refer the reader to [39] for details. Following [39], we constrain the computation of the loss L_{disc} to the dilated road regions \mathcal{R} . This speeds up convergence in the early stages of the training, when path minima may be found

far away from true roads.

Penalizing False Connections

We could take L_{TOPO} to simply be L_{disc} but we have observed that this results in many false positive road segments and that this behavior is difficult to counteract only by balancing the regression and connectivity losses with the coefficient α in Eq. 2.2. To remedy this, we also enforce connectivity of background regions, which prevents false positive roads, as

$$L_{\text{conn}}(y, \hat{y}) = \sum_{A \in \mathcal{B}} \sum_{p \in A} \nu_p (\hat{y}[p] - y_D[p])^2, \quad (2.8)$$

where ν_p is the number of pairs of pixels $q, r \in A$, for which p is the smallest pixel on the maximin path between q and r , and is computed similarly to w_p , and $y_D[p]$ is the value of the ground truth distance map at pixel p . Like L_{disc} , the formulation of L_{conn} has been used before in [39], and our innovation here is that we apply both of these losses to background regions, in order to indirectly enforce the correct connectivity of foreground roads and canals.

Introducing Sliding Windows.

We can compute L_{TOPO} as described above on the whole image. However, when we do that, almost all pixels are assigned weights $w = 0$ in Eq. 2.6 and a single road pixel gets a weight equal to the product of the size of the connected components that the road should separate. This is because, in the presence of an evident road interruption, all maximin paths go through this interruption. This might seem desirable in theory, but in practice it makes learning unstable. Since only a small minority of pixels generate extremely large gradients, no error signal is distributed among the remaining ones.

To overcome this problem we compute L_{TOPO} independently for 64×64 image patches that cover the image, and sum the results. This ensures that at least one road pixel per window is taken into account and that its weight is not larger than $N^2/4$, where N is the number of pixels in the window. As shown in Fig. 2.3, this also lets us handle dead-ending roads that do not separate the global map into disjoint areas.

2.4 Experiments

We now describe the dataset we have tested our approach on, the baselines to which we compare our results, and the metrics we used to assess the quality of the reconstructions. We then demonstrate that our new loss improves the results of networks that rely solely on conventional losses and substantially outperform recently proposed road reconstruction methods.

2.4.1 Datasets

We performed experiments on three publicly available datasets.

- *RoadTracer*. A recently published dataset of high-resolution satellite images covering urban areas of forty cities in six different countries [6]. As in [6, 62, 103, 70], fifteen cities are used as a test set, and the remaining twenty five as a training set. The ground truth was generated using OpenStreetMap.
- *DeepGlobe*. Aerial images of rural areas in Thailand, Indonesia and India [30]. The dataset comprises around 8500 images. For a fair comparison to [7], we use the same split with 4695 training and 1530 test images, but no validation images.
- *Canals*. Aerial images of water drainage canals in rural areas of Malaysia [87]. The dataset comprises a single 9768×10718 ortho-photo. We use a crop of size 2630×1576 pixels for testing and the rest for training purposes.
- *Massachusetts*. The Massachusetts dataset [68] features both urban and rural neighborhoods, with many different kinds of roads ranging from small paths to highways. For a fair comparison to [48], we split the data into three equal folds and performed a three-way cross validation.

Together, these datasets exhibit a very large variation of terrain type, which makes them an exhaustive benchmark for aerial road and drainage canal network reconstruction.

2.4.2 Baselines

We compare the results of our algorithm to the following state-of-the-art methods.

- *Segmentation*. A baseline algorithm from [6], combining segmentation, thresholding, skeletonization, and conversion of the skeleton to a graph. Road network reconstructions for the RoadTracer dataset were made available online by the authors [5].
- *RoadTracer*. Iterative graph construction where node locations are selected by a CNN [6]. The road network reconstructions were released publicly by the authors [5].
- *Seg-Path*. A unified approach to segmenting linear structures and classifying potential connections [70]. The road network reconstructions were provided to us by the authors.
- *RCNNU-Net*. Recursive image segmentation with post-processing for graph extraction [103]. The authors provided the probability maps.
- *DeepRoad*. Image segmentation followed by post-processing focused on fixing missing connections [66]. The graphs for the RoadTracer dataset were published by the authors of this data set [5].

Chapter 2. Promoting Connectivity of Network-like Structures by Enforcing Region Separation

- *PolyMapper*. Reconstructing a map by sequential construction of closed polygons [62]. The graphs were provided to us by the authors.
- *MultiBranch*. A recursive architecture co-trained in road segmentation and orientation estimation [7]. To obtain the road network reconstructions, we trained the network using the code published by the authors.
- *LinkNet*. An encoder-decoder architecture [21] co-trained in segmentation and orientation estimation [7]. We trained the network using the code made available by the authors.
- *PersHomo*. It uses a loss function based on the concept of persistent homology from mathematical topology [48]. The results for the *Massachusetts* data set were provided by the authors.
- *UNet*. Our own implementation of U-Net [81] trained with mean squared error.
- *DRU*. A recurrent U-Net iteratively refining segmentation output [98], trained by us with the mean squared error.
- *MALA*. A recent implementation of the MALIS loss to enforce the correct connectivity when segmenting electron microscopy images of neural tissue [39].

2.4.3 Network Architecture and Training Details

We compare these baselines against three variants of our approach introduced in Section 4.3.

- *UNet + TOPO-glo*. A U-Net, trained with our connectivity loss computed in the full image.
- *UNet + TOPO-win*. A U-Net, trained with our loss computed in windows of size 64×64 pixels.
- *DRU + TOPO-win*. A recurrent U-Net [98], trained with the windowed version of our loss.

In the *UNet* experiments, we used the standard U-Net [81] architecture, with five blocks, each with three sequences of convolution-ReLU-batch normalization. Max-pooling in 2×2 windows followed each of the blocks. The initial feature size was set to 32 and grew to 1024 in the smallest feature map in the network. We augmented the input data with vertical and horizontal flips and random rotations.

In the experiments with *DRU*, we used a recurrent U-Net with the same architectural features that we used in *UNet* experiments. There is a dual-gated recurrent unit in the bridge part of the network [98]. During training, we used three recurrent iterations. After each recurrent iteration, the output of the network is used as an additional channel to the input of the next

iteration. For the first iteration, this additional channel is set to 0. During inference, we used the output of the second iteration which produced the best results.

We trained the network with the ADAM algorithm [54], with the learning rate set to $1e-4$. We set the coefficients $\alpha = 1e-3$ in Eq. 2.2 and $\beta = 0.1$ in Eq. 2.4. We justify these choices in Section 2.4.6.

2.4.4 Performance Measures

Comparing connectivity of road reconstructions is difficult, because the reconstructions rarely overlap with the ground truth, and often deviate from it significantly. There seems to be no consensus concerning the single best evaluation technique in the existing literature – we have found five different connectivity-oriented metrics in concurrently published recent work. To provide exhaustive evaluation, we used all of them in our experiments.

- *APLS* Average Path Length Similarity, defined as an aggregation of relative length difference of shortest paths between pairs of corresponding points in the ground truth and predicted maps [92].
- *TLTS* Statistics of lengths of shortest paths between corresponding pairs of end points randomly selected in the predicted and ground-truth networks [99]. We report the fraction of paths where the relative length difference is within 5%.
- *JCTA* junction score, evaluating the number of roads intersecting at each junction [6]. Consists of road recall, averaged over the intersections of the ground-truth and road precision, averaged over the intersections of the prediction. We report the corresponding F1 score.
- *HM* Compares the sets of graph locations accessible by traveling away from randomly chosen pairs of corresponding points in both graphs [10]. We report the corresponding F1-score.
- *CCQ* To complement the connectivity-oriented evaluation, we also computed the most popular metric that measures spatial co-occurrence of annotated and predicted road pixels, rather than connectivity. The Correctness, Completeness and Quality are equivalent to precision, recall and intersection-over-union, where the definition of a true positive has been relaxed from spatial coincidence of prediction and annotation to co-occurrence within a distance of 5 pixels [100]. We report the Quality as our single-number metric.

2.4.5 Comparative Results

We report the performance of our method on the *RoadTracer*, *DeepGlobe*, *Canals*, and *Massachusetts* datasets in Tabs. 2.1, 2.2, 2.3, and 2.4. For our own approach, we report an average

Chapter 2. Promoting Connectivity of Network-like Structures by Enforcing Region Separation

Table 2.1: Comparative results on the *RoadTracer* dataset [6]. Our loss function makes even the simple *UNet* attain state of the art performance. Computing the loss in windows results in improvement of four out of five performance criteria. The results for our method are means and standard deviations over three independent training runs.

Method	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>HM</i>	<i>CCQ</i>
<i>Segmentation</i>	62.5	33.0	78.2	69.4	54.4
<i>RoadTracer</i>	59.1	40.6	81.2	70.5	47.8
<i>Seg-Path</i>	68.1	46.5	75.4	67.6	54.0
<i>RCNNU-Net</i>	48.2	18.4	75.9	68.8	62.8
<i>DeepRoad</i>	24.6	6.4	51.4	46.8	43.6
<i>PolyMapper</i>	61.3	31.5	80.0	53.7	35.7
<i>UNet</i>	66.3	40.0	77.5	68.2	59.3
<i>MALA</i>	68.6	42.8	77.8	65.8	62.7
<i>UNet+ TOPO-glo</i>	71.9±1.7	46.2±2.1	84.7±2.3	70.8±1.2	63.5±1.8
<i>UNet+ TOPO-win</i>	75.4±1.6	49.6±1.4	82.6±0.6	75.9±1.5	68.4±0.4

Table 2.2: Comparative results on the *DeepGlobe* dataset. Our loss function improves the performance of both *UNet* and *DRU* in terms of all the metrics, with *DRU* attaining the state-of-the-art performance. The results for our method are again means and variances over three independent training runs.

Method	Connectivity-oriented				pixel-based
	<i>APLS</i> ↑	<i>TLTS</i> ↑	<i>JCT</i> ↑	<i>HM</i> ↑	<i>CCQ</i> ↑
<i>LinkNet</i>	67.7	60.6	66.2	73.4	77.2
<i>MultiBranch</i>	70.8	65.2	71.1	75.6	79.4
<i>UNet</i>	62.3	59.9	66.4	72.7	68.8
<i>DRU</i>	75.2	65.4	67.2	76.6	80.1
<i>UNet+ TOPO-win</i>	75.4±1.4	70.3±0.7	71.3±0.8	80.1±0.3	77.4±1.9
<i>DRU+ TOPO-win</i>	77.3±0.8	68.3±0.3	71.6±0.4	79.4±0.3	80.5±0.5

over three independent runs along with the corresponding standard deviations. The resulting delineations are depicted qualitatively in Figs. 2.4, 2.5, 2.7, and 2.6. On average, *DeepGlobe* features simpler roads with fewer opportunities for mistakes than *RoadTracer* and *Canals*.

As can be seen in Table 2.1, on the *RoadTracer* dataset, the non-windowed version of our approach *UNet+ TOPO-glo* performs very well compared to all the baselines and the windowed version *UNet+ TOPO-win* clearly outperforms *all* the baselines on *all measures*, save one. The only exception is CCQ on the *DeepGlobe* dataset, in Tab. 2.2, which we attribute to the fact that our loss is designed to enforce connectivity and CCQ does not measure it. This is remarkable because many of the competing architectures rely on far more sophisticated networks than

Table 2.3: Comparative results on the *Canals* dataset. Our loss function boosts the performance of both *UNet* and *DRU* in terms of all the five metrics. The results for our method are means and standard deviations over three independent training runs.

Method	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>HM</i>	<i>CCQ</i>
<i>LinkNet</i>	71.5	26.8	75.3	57.8	83.4
<i>MultiBranch</i>	77.6	33.1	77.5	63.2	84.2
<i>UNet</i>	70.9	30.3	76.4	61.4	81.4
<i>DRU</i>	71.4	32.7	76.9	62.1	80.3
<i>UNet+ TOPO-win</i>	76.2±0.1	35.8±0.1	79.3±0.3	63.5±0.1	84.9±0.3
<i>DRU+ TOPO-win</i>	78.3±0.1	43.0±0.4	78.7±0.4	66.9±0.3	84.7±0.2

Table 2.4: Comparative results on the on the *Massachusetts* dataset. Using our loss function combined with *UNet* outperforms [48] on all metrics. For a fair comparison, we used the same *UNet* architecture as in [48].

Method	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>HM</i>	<i>CCQ</i>
<i>PersHomo</i>	58.9	36.8	73.9	60.7	62.3
<i>UNet+ TOPO-win</i>	73.4±3.6	53.2±4.4	81.4±1.9	69.9±2.6	65.3±3.2

the simple U-Net we use. This demonstrates that our topological loss function does the job it was designed to do. In particular, the fact that we outperform *MALA*, which enforce road connectivity by directly using the MALIS loss, confirms our earlier claim that enforcing dis-connectivity is more effective for complex road networks.

As can be seen in Tables 2.2 and 2.3, *UNet+TOPO-win* also outperforms the baselines on *DeepGlobe* and *Canals*. When we replace the simple U-Net by its more sophisticated *DRU* version, the performance tends to improve further on most measures but not all, which shows that our MALIS-based loss function can also boost the performance of more sophisticated architectures.

To further confirm this effectiveness of our loss function, we compare in Tab. 2.4 our results to those obtained by *PersHomo* on the *Massachusetts* dataset, which were given by the authors of [48]. For a fair comparison, we used the same network implementation as them for these experiments. It features two times fewer features in each layer than the *UNet* we used in all other experiments. We also used the same data splits. We consistently outperform *PersHomo*, mainly in *TLTS* and *APLS*. As illustrated in Fig. 2.7, this is due to the small disconnections that *PersHomo* produces, which our loss suppresses more effectively.

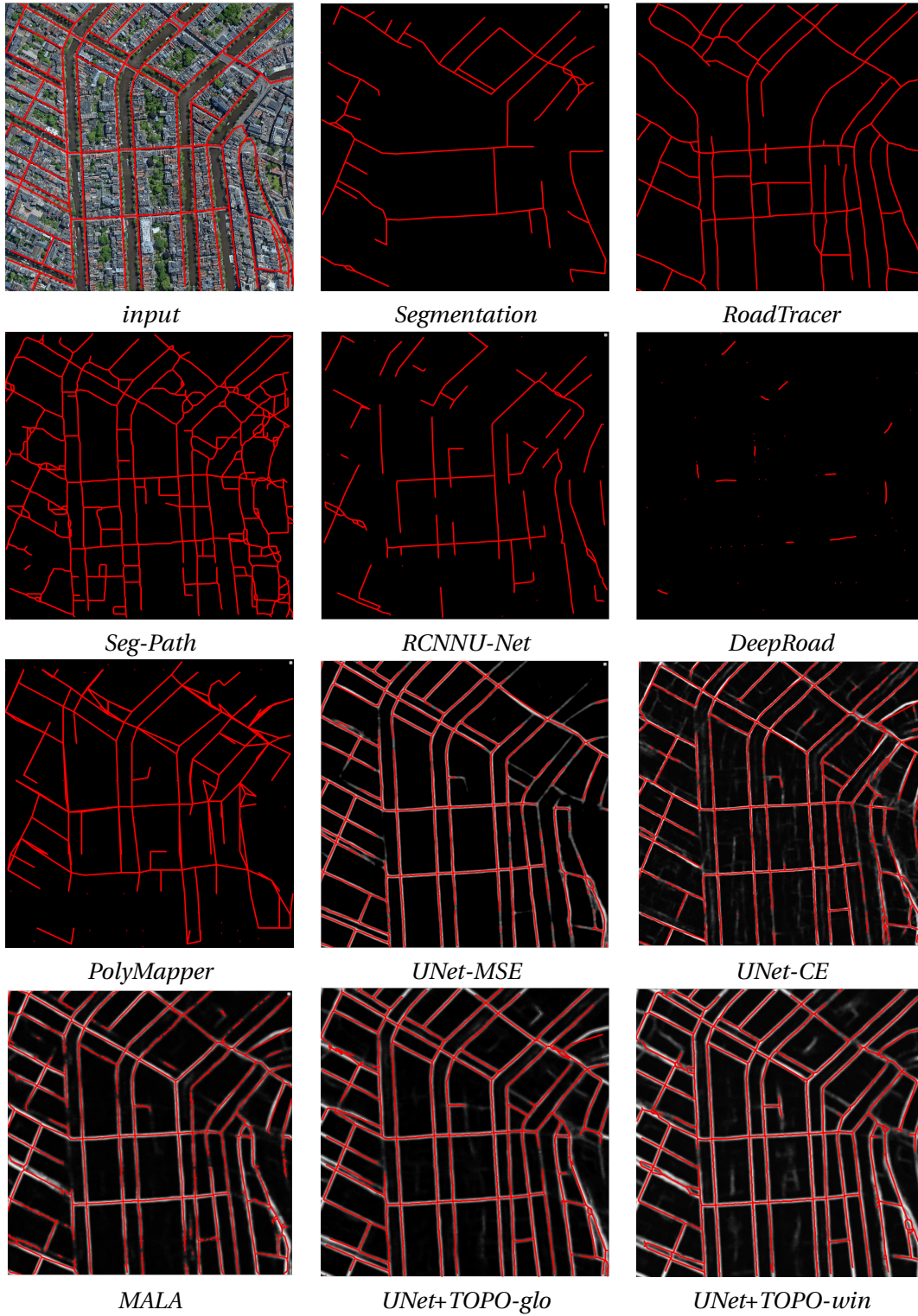


Figure 2.4: Comparative results on the *RoadTracer* dataset. For our results, we overlaid the graphs on the inferred distance maps.

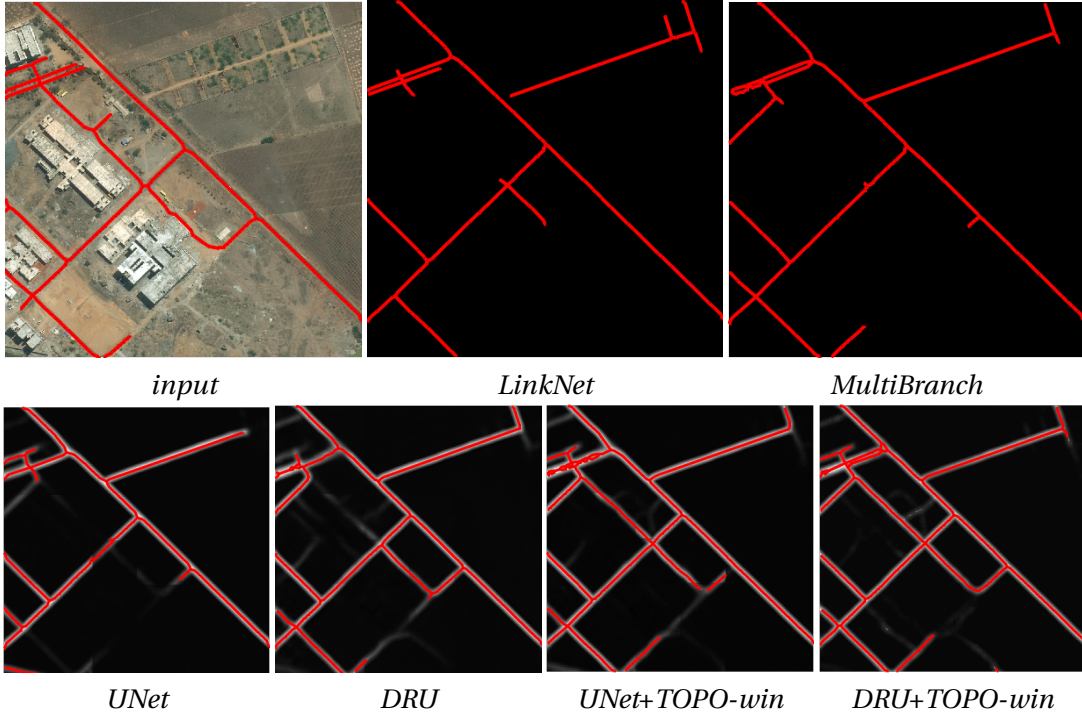


Figure 2.5: Comparative results on the *DeepGlobe* dataset. For the results of our method, we overlaid graphs on the inferred distance maps.

2.4.6 Ablation Study

We run a number of ablation studies to investigate the impact of the hyper-parameters of our method on performance.

Training Times

The performance gain delivered by our method comes at the cost of increased training time. In the experiments with *UNet*, the time needed for a single training iteration grew from 1.12s using only the simple MSE loss to 2.51s when using our MALIS-based loss. However, as the testing procedure remains unchanged, there is no slowdown at test time.

Varying the Impact of the Connectivity Loss

Our loss function defined in Eq. 2.2 is the sum of the mean square error L_{MSE} and the connectivity term L_{TOPO} weighted by a coefficient α . In order to evaluate the impact of changing α on performance, we trained our *UNet* with different α on the standard split of the *RoadTracer* dataset. We report the results in Tab. 2.5. Qualitative results are shown in Fig. 2.8. Setting α too low or too high adversely affects performance and its optimal value is in the order of $1e-3$. The explanation of this phenomenon is provided in Fig. 2.8. For low values of α , the effect of

Chapter 2. Promoting Connectivity of Network-like Structures by Enforcing Region Separation

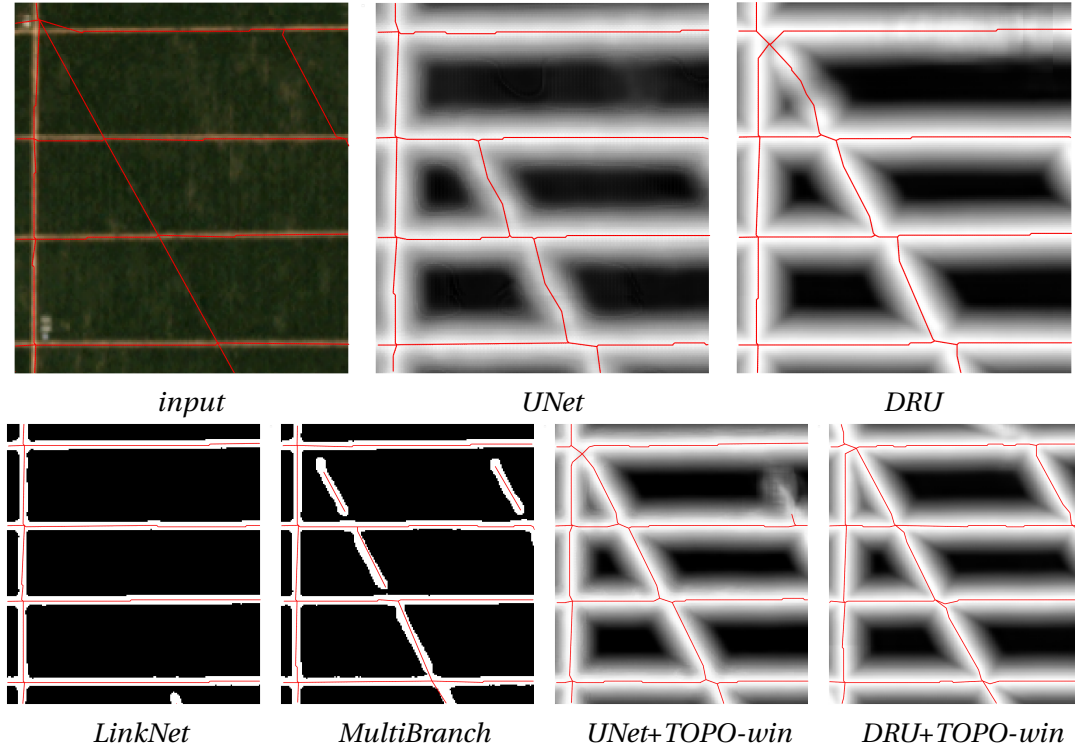


Figure 2.6: Comparative results on the *Canals* dataset. For the results of our method, we overlaid graphs on the inferred distance maps.

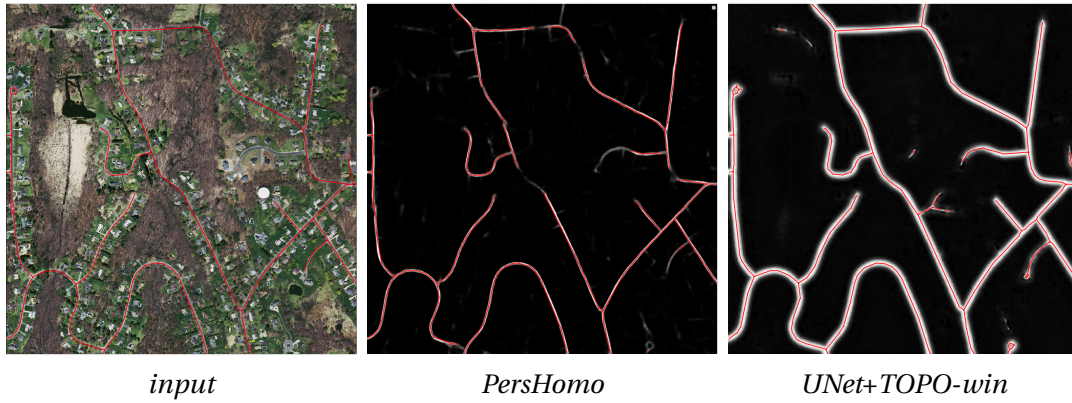


Figure 2.7: Qualitative results on the *Massachusetts* dataset. For our method, we overlaid graphs on the inferred distance maps.

the connectivity-oriented component of the loss function is negligible. When α is increased, more and more connections are represented in the distance map. However, when α is set very high, the network starts to privilege disconnecting background image regions, even with no obvious roads in the input, creating false positive road segments.

Table 2.5: Impact of changing the value α that balances L_{MSE} against L_{TOPO} in Eq. 2.2, using *UNet+TOPO-win* on the *RoadTracer* dataset. The window size is fixed to 64x64 and β to 0.1. The corresponding results are depicted qualitatively by Fig. 2.8.

α	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>HM</i>	<i>CCQ</i>
1e-2	72.3	46.3	80.3	73.1	66.5
1e-3	75.8	49.7	82.8	76.0	68.6
1e-4	71.4	45.9	81.9	73.4	67.1
0.0	66.3	40.0	77.5	68.2	59.3

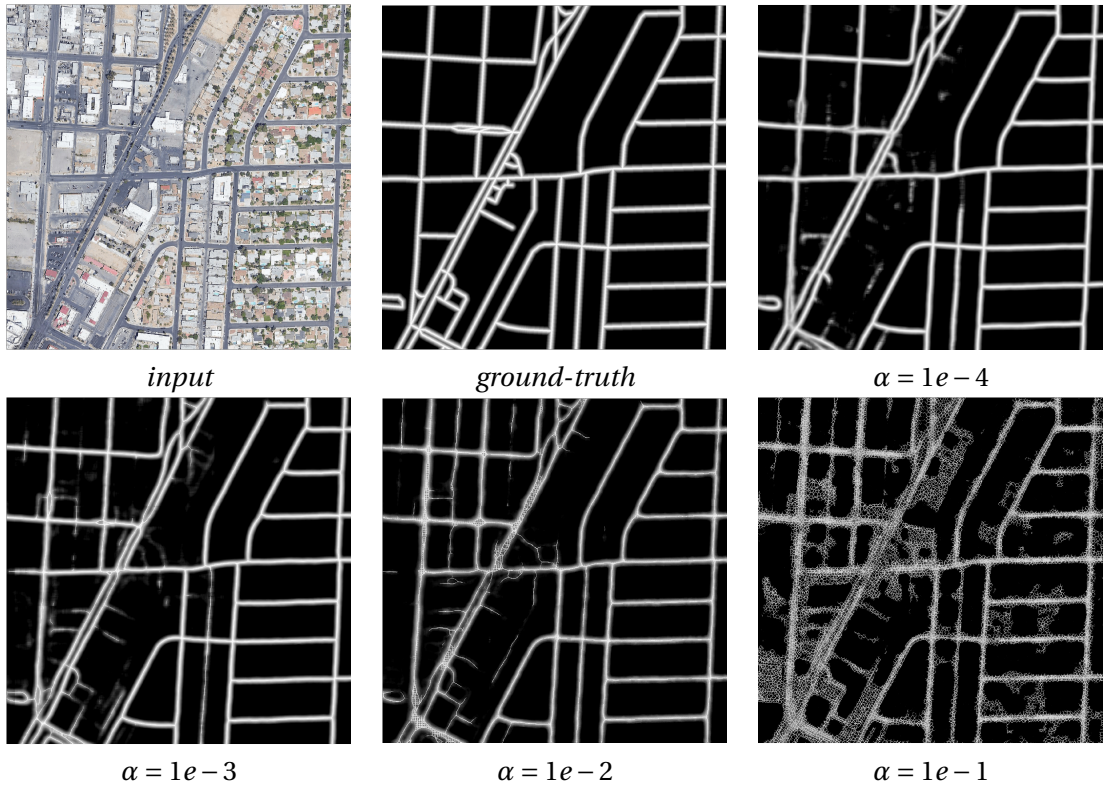


Figure 2.8: Effect of changing α in Eq. 2.2 on the distance map the neural network outputs. As α increases, the road map becomes more complete until α becomes so large that it promotes spurious connections even where no roads are present in the image.

Balancing Connectivity versus Dis-Connectivity

In Eq. 2.4, we introduced a loss term L_{conn} to prevent false positives and it is weighted by parameter β . In the top part of Tab. 2.6, we report results obtained by varying β . Qualitative results are shown in 2.9. The best ones are obtained for $\beta = 0.1$, meaning that the term preventing disconnections has ten times more impact on the loss than the term preventing false positive roads, which simply suggests that a balance between these two terms is required.

Chapter 2. Promoting Connectivity of Network-like Structures by Enforcing Region Separation

Table 2.6: Impact of changing the value of β that balances the connectivity and dis-connectivity components of our loss in Eq. 2.4, using *UNet+TOPO-win* on the *RoadTracer* dataset. The corresponding results are depicted qualitatively by Fig. 2.9. The bottom part of the table features the performance measures obtained when *not* using the L_{MSE} loss. They are lower, which shows that balancing the connectivity and dis-connectivity components of L_{TOPO} is not enough to ensure maximum performance.

L_{MSE} used?	β	Connectivity-oriented				pixel-based
		<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>HM</i>	<i>CCQ</i>
yes	1e-0	71.4	43.8	80.9	73.2	66.5
	1e-1	75.8	49.7	82.8	76.0	68.6
	1e-2	74.3	46.2	79.5	74.5	65.3
no	1e-0	60.6	35.2	71.0	62.3	56.4
	1e-1	64.7	39.9	75.3	64.5	57.3
	1e-2	61.2	36.6	72.4	61.9	54.9

Importance of L_{MSE}

L_{TOPO} has both a component enforcing road connectivity and one preventing false positive road predictions. Hence, one might ask if the distance regression term L_{MSE} is still needed. To verify this, we ran experiments on the *RoadTracer* dataset. We turned off the L_{MSE} term and again varied β to balance the connectivity and dis-connectivity terms of L_{TOPO} . The results are presented in the bottom part of Tab. 2.6 and are clearly less good than those we obtained when using L_{MSE} . We attribute this to the fact that the gradient of L_{TOPO} is sparse, even when the loss is evaluated only in windows. When the two losses are used together, L_{MSE} generates dense gradients that are particularly useful away from the roads while L_{TOPO} provides sparse gradients focused on ensuring the correct connectivity.

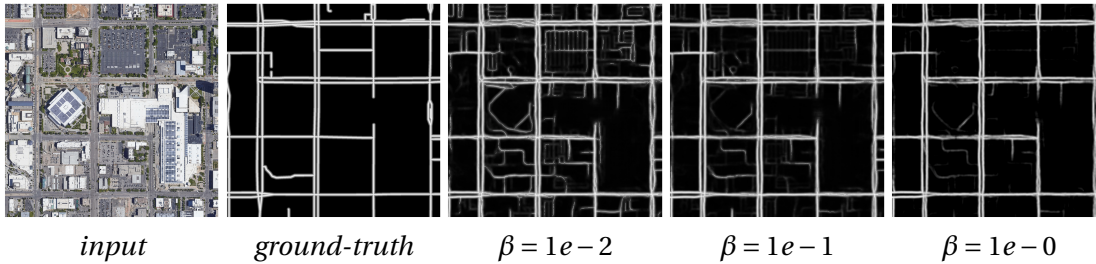


Figure 2.9: Effect of changing β in Eq. 2.4 on the distance map the neural network outputs. As β is increases, the predictions become more precise. It reduces false positive roads until β becomes so large that it creates disconnections on actual roads.

Varying the Window Size

The third, and last, hyper-parameter of our method is the window size. Computing the loss in windows, or image crops, as opposed to globally in the entire image, has the advantage of preventing accumulating all the error signal in a single pixel. The smaller the window, the more evenly the gradient is distributed among road pixels. The windowed version of the loss also enables enforcing connectivity of dead-ending roads, as small windows are often subdivided even by dead-ending roads. Large window sizes do not have this effect, as roads shorter than the window size end in the middle of the window, without splitting it into disjoint tiles. To discover the optimal window size, we tested its effect on performance. The results, presented in Tab 2.7 and 2.8, confirm that mid-size windows work best. Setting the window size to 64×64 pixels resulted in the highest performance, and increasing or decreasing the window decreases performance.

Table 2.7: The impact of window size on performance. Results of experiments on the *Road-Tracer* dataset. α is fixed to $1e-3$ and β to 0.1. *UNet+TOPO-win* is used in all experiments.

Window Size	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>HM</i>	<i>CCQ</i>
(16x16)	68.3	39.2	79.2	67.4	59.4
(32x32)	72.1	45.8	78.9	72.7	65.7
(64x64)	75.8	49.7	82.8	76.0	68.6
(128x128)	76.1	46.4	81.7	74.5	68.3

Table 2.8: The impact of changing the window size on performance. Results of experiments on the *DeepGlobe* dataset. α is fixed to $1e-3$ and β to 0.1. *UNet+TOPO-win* is used in all experiments.

Window Size	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>HM</i>	<i>CCQ</i>
(32x32)	74.1	67.3	65.2	73.4	74.8
(64x64)	75.2	69.8	71.2	79.8	77.0
(128x128)	74.3	68.2	72.0	79.6	77.2

Comparing Mean Squared Error with Cross Entropy

Our loss function combines a connectivity-oriented term with mean squared error. This combination outperforms a number of existing networks, trained with cross entropy. We therefore investigated if just switching from the more common cross entropy to mean squared error, without our connectivity-oriented loss, impacts the performance. We present the results in Tab. 2.9. We conclude that solely switching from pixel classification to distance map estimation does not warrant the increased connectivity, and it is our connectivity-oriented

Chapter 2. Promoting Connectivity of Network-like Structures by Enforcing Region Separation

Table 2.9: Comparison of Cross Entropy and Mean Square Error. Results of experiments on the *RoadTracer* dataset [6].

Method	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>HM</i>	<i>CCQ</i>
<i>UNet-CE</i>	60.4	30.6	79.2	74.2	63.3
<i>UNet-MSE</i>	66.3	40.0	77.5	68.2	59.3
<i>UNet+TOPO-glo</i>	72.5	46.3	84.7	70.3	63.8
<i>UNet+TOPO-win</i>	75.8	49.7	82.8	76.0	68.6

term that does it.

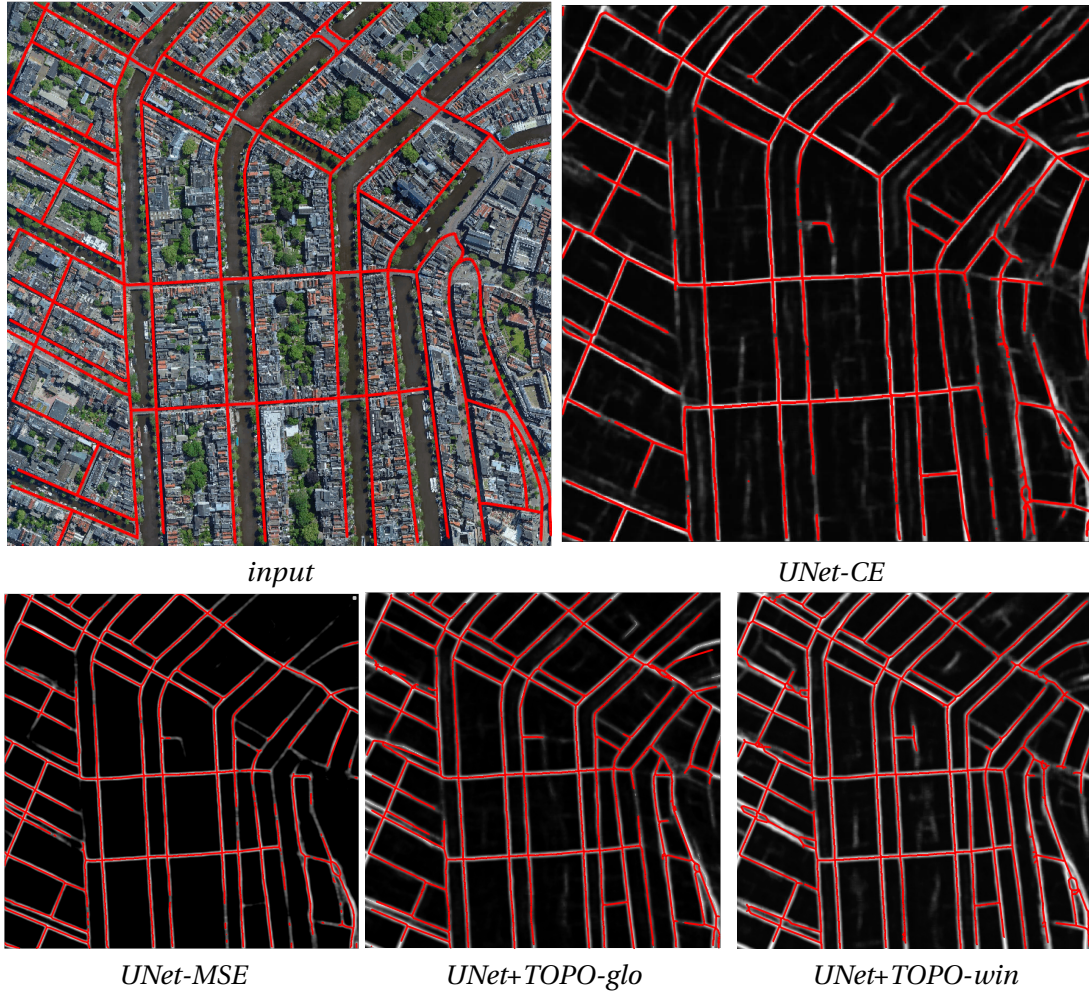


Figure 2.10: A comparison between the results obtained with the CE and MSE losses, on the *RoadTracer* data set. For the results, we overlaid graphs on the inferred distance maps.

2.5 Conclusion

We have introduced a differentiable loss function that effectively enforces proper connectivity on the output of binary segmentation ConvNets for the purpose of road network delineation. Using this loss function to train a simple U-Net allows us to outperform far more sophisticated architectures on challenging benchmark datasets. This suggests that we may not yet have unleashed the full power of these simpler networks and that adding appropriate constraints during training might be a way to do so.

We have so far limited ourselves to networks of roads and drainage canals, but networks of linear structures are also pervasive in biomedical 3D imagery. They range from neural structures to blood vessels and many others. In future work, we will therefore expand our approach to handle 3D image stacks and address a much broader range of applications. The difficulty we will have to overcome stems from the fact that, in 2D, linear structures cut the image into disjoint regions, which is not the case anymore in 3D. Fortunately, we have demonstrated in earlier work [55, 56] that it is possible to effectively train a 3D network using only 2D annotations in planar volume slices in which the method described in this chapter applies.

3 Enforcing Connectivity of 3D Linear Structures Using Their 2D Projections

Many biological and medical tasks require the delineation of 3D curvilinear structures such as blood vessels and neurites from image volumes. This is typically done using neural networks trained by minimizing voxel-wise loss functions that do not capture the topological properties of these structures. As a result, the connectivity of the recovered structures is often wrong, which lessens their usefulness. In this chapter, we propose to improve the 3D connectivity of our results by minimizing a sum of topology-aware losses on their 2D projections. This suffices to increase the accuracy and to reduce the annotation effort required to provide the required annotated training data. This work appeared in [76].

Oner, D., Osman, H., Kozinski, M., and Fua, P. *Enforcing Connectivity of 3D Linear Structures Using Their 2D Projections*. In Conference on Medical Image Computing and Computer Assisted Intervention – MICCAI 2022.

3.1 Introduction

Delineating 3D curvilinear structures, such as veins and arteries visible in computed tomography (CT) scans, or dendrites and axons revealed by light microscopy (LM) scans, is central to many applications. State-of-the-art algorithms typically rely on deep networks trained to classify each voxel as either foreground or background by minimizing a voxel-wise loss. Networks trained this way are good at voxel classification but nevertheless prone to topological errors, such as unwarranted gaps in the linear structures and false interconnections between them. This mostly occurs when vessels and neuronal projections appear as thin but densely woven structures and misclassifying a few voxels can disrupt their connectivity without much influence on voxel-wise accuracy. These errors greatly reduce the usefulness of the resulting arborization models. Correcting them requires manual interventions, which is very time consuming when performed on whole-brain microscopy scans or whole-organ CT scans, especially at scales sufficiently large to produce statistically significant results.

In other words, networks trained by minimizing losses such as the Cross Entropy and the

Mean Squared Error, which are sums of per-voxel terms *independent* of all other voxels, struggle to learn patterns formed *jointly* by groups of voxels [49, 27, 75]. A promising approach to addressing this issue is to develop *topology-aware* loss functions that evaluate patterns emerging from predictions for multiple voxels. This includes perceptual losses [71], loss functions based on persistent homology [49, 27], and a loss that enforces continuity of linear structures by penalizing interconnections between background regions on their opposite sides [75]. The latter has proved to be more effective for 2D images than the others but does not naturally generalize to 3D volumes.

In this chapter, we start from the observation that continuity of a 3D linear structure implies continuity of its 2D projections. Hence, we can use a topology-aware loss, such as the one of [75], to penalize connectivity errors in 2D projections of the 3D predictions, thereby indirectly penalizing the errors in the 3D originals. This also means that we can use 2D annotations, which are much easier to obtain than full 3D ones, to train a 3D network. This is close in spirit to the approach of [56] in which delineation networks are trained by minimizing a loss function in maximum intensity projections of the predictions and the annotations.

We demonstrate the effectiveness of our approach for delineating neurons in light microscopy scans and tracing blood vessels in Magnetic Resonance Angiography scans. Not only do we produce topologically correct delineations, but we also reduce the annotation effort required to train the networks.

3.2 Related Work

3.2.1 Delineation of 3D linear Structures

Over the years, many approaches to delineating 3D linear structures have been proposed. They range from hand-designing filters that are sensitive to tubular structures [36, 58, 89] to learning such filters [102, 12] using support vector machines [50], gradient boost [85], or decision trees [90].

Neural networks have now become the dominant technique [69, 42, 63, 78, 45, 101]. They are often trained by minimizing pixel-wise loss functions, such as the cross-entropy or the mean square error. As a result, the delineations they produce often feature topological mistakes, such as unwarranted gaps or false connections. This occurs because it often takes very few mislabeled pixels to significantly alter the topology with little impact on the pixel-wise accuracy.

3.2.2 Topology-Aware Loss Functions

Specialized solutions to this problem have been proposed in the form of loss functions comparing the topology of the predictions to that of the annotations. For example, the perceptual loss [70] has been shown to be sensitive to topological differences between the prediction and

the ground truth, but cannot be guaranteed to penalize all of them. Persistent Homology [32] is an elegant approach to describing and comparing topological structures. It has been used to define topology-oriented loss functions [48, 27, 16]. Unfortunately, computing this loss is computationally intensive and error-prone because it does not account for the location of topological structures. cIDice [83] is a loss function that employs a soft skeletonization algorithm to compare the topology of the prediction and the annotation, but it is designed for volumetric segmentation, whereas we focus on tracing linear structures given their centerlines.

For delineation of 2D road networks, existing approaches are outperformed by the method of [75] that repurposes the MALIS loss initially proposed to help better segment electron microscopy scans [13, 39] to improve the topology of reconstructed loopy curvilinear networks. Unfortunately, the algorithm of [75] can only operate in 2D. In this paper, we show how it can nevertheless be exploited for 3D delineation in volumetric images.

3.3 Method

We train a deep network to regress the distance from each voxel of the input 3D image x to the center of the nearest linear structure. We denote the predicted 3D distance map by y . The annotations are given in the form of a graph with nodes in 3D space. We denote the set of edges of this graph by \mathcal{E} . From the annotation graph, we compute the truncated ground truth distance map \hat{y} . For a voxel p , $\hat{y}[p] = \min((\min_{\epsilon \in \mathcal{E}} d_{p\epsilon}), d_{\max})$, where $d_{p\epsilon}$ is the distance from p to the annotation edge ϵ , and d_{\max} is the truncation distance set to 15 pixels.

A simple way to train our deep net is to minimize a Mean Squared Error loss $L_{\text{MSE}}(y, \hat{y})$ for all training images. As discussed in Section 3.2, minimizing such a voxel-wise loss does not guarantee that connectivity is preserved because mislabeling only a few voxels is enough to disrupt it. For 2D images, this problem has been addressed with a loss term L_{TOPO} that effectively enforces continuity of 2D linear structures [75]. Unfortunately, it is limited to 2D data by design and cannot be extended to 3D. To bypass this limitation, we leverage the observation that continuity of 3D structures implies continuity of their 2D projections and evaluate L_{TOPO} on 2D projections of the 3D predicted and ground truth distance maps. We introduce the 2D connectivity-oriented loss term and the technique to train 3D deep networks on 2D projections in the following subsections.

3.3.1 Connectivity loss

In this section, we recall the intuition behind the connectivity-oriented loss term L_{TOPO} of [75]. We refer the reader to the original publication for a more detailed explanation. As illustrated by Fig. 3.1(a), a path connecting pixels on opposite sides of a linear structure must cross that structure and should therefore contain at least one pixel p such that the predicted distance map $y[p] = 0$. If y contains erroneous disconnections, then it is possible to construct a path that connects pixels on opposite sides of the structure, but only crosses pixels with

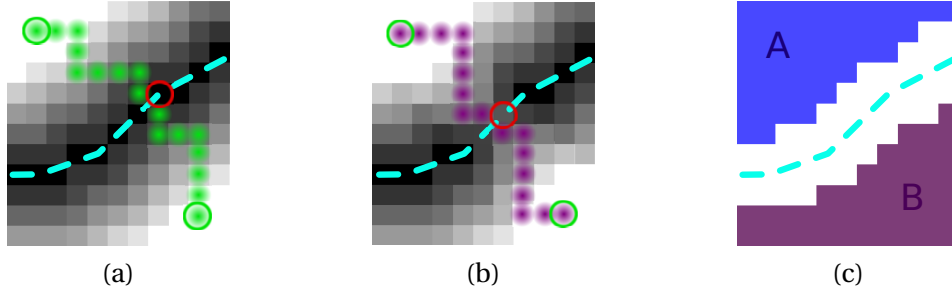


Figure 3.1: The intuition behind L_{TOPO} . (a) In a perfect distance map, any path connecting pixels on the opposite sides of an annotation line (dashed, magenta) crosses a zero-valued pixel (red circle). (b) If a distance map has erroneously high-valued pixels along the annotation line, the maximin path (violet) between the same pixels crosses one of them (red circle). (c) The connectivity-oriented loss L_{TOPO} is a sum of the smallest values crossed by maximin paths connecting pixels from different background regions. The background regions are computed by first dilating the annotation (dilated annotation shown in white), to accommodate possible annotation inaccuracy.

predicted distance values larger than zero, as depicted by Fig. 3.1(b). In particular, a *maximin* path, that is, the path with largest smallest pixel among all possible paths between the same end points, is guaranteed to pass through an interruption of the linear structure, if it exists. L_{TOPO} minimizes the smallest pixel on the maximin path between each pair of end points that belong to background regions on the opposite sides of annotated linear structures, shown in Fig. 3.1(c). It has proven effective in enforcing connectivity of 2D linear structures, but cannot be extended to 3D, because 3D linear structures do not subdivide 3D volumes into disjoint background regions.

3.3.2 Projected Connectivity Loss

The key observation underlying our approach is that 3D continuity of three-dimensional curvilinear structure, represented as a depth-map, implies its continuity in 2D minimum-intensity projections of the depth map. The reverse is not true: a projection of a discontinuous 3D depth-map might appear continuous if it is taken along the direction tangent to the linear structure at discontinuity. However, even in such case, the discontinuity appears in other projections, taken along directions orthogonal to the direction of the first projection, as shown in Fig. 3.2. In general, given three orthogonal projections of a 3D volume, each discontinuity appears in at least two of them, unless it is occluded by other linear structures. Hence, we evaluate the topology-enforcing loss L_{TOPO} on projections of the predicted and ground truth distance maps along the principal directions. Let y^i be the min-intensity projection of y along direction i , where i can be one of the axes x , y , or z and the corresponding projection of \hat{y} be \hat{y}^i . We take our connectivity-enforcing loss to be

$$L_{\text{conn}}(y, \hat{y}) = \sum_{i \in \{x, y, z\}} L_{\text{TOPO}}(y^i, \hat{y}^i), \quad (3.1)$$

where L_{TOPO} is the 2D connectivity loss of [75] discussed above. This loss can easily be differentiated with respect to the values of y , as the minimum-intensity projection is just a column-wise *min* operation.

3.3.3 Total Loss

The total loss that we minimize can therefore be written as

$$L_{3D}(y, \hat{y}) = L_{\text{MSE}}(y, \hat{y}) + \alpha L_{\text{conn}}(y, \hat{y}), \quad (3.2)$$

where α is a scalar that weighs the influence of the two terms. As discussed above, L_{MSE} can be simply computed as the mean squared difference between the predicted and ground truth 3D distance maps.

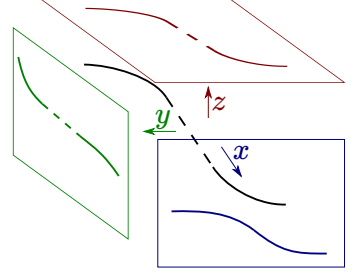


Figure 3.2: Disconnections in 3D linear structures appear in at least two out of three orthogonal projections, unless the structure is occluded.

This is a perfectly valid choice when 3D annotations are available, but such annotations are typically hard to obtain. Fortunately, it has been shown in [56] that one can train a network to perform 3D volumetric delineation given *only* 2D annotations in Maximum Intensity Projections. This saves time because manually delineating in 2D is much easier than in 3D. Since we impose our connectivity constraints on projections along the axes x , y , and z , it makes sense to also provide annotations only for the corresponding projections of the input volume x , and generate from them ground truth distance maps \hat{y}_x , \hat{y}_y , and \hat{y}_z . To replace the 3D ground truth \hat{y} , that L_{3D} requires, we can rewrite our total loss as

$$L_{2D}(y, \hat{y}_x, \hat{y}_y, \hat{y}_z) = \sum_{i \in \{x, y, z\}} L_{\text{MSE}}(y^i, \hat{y}_i) + \alpha \sum_{i \in \{x, y, z\}} L_{\text{TOPO}}(y^i, \hat{y}_i), \quad (3.3)$$

where the Mean Squared Error is evaluated on the minimum-intensity projections of the predicted distance map and the distance map produced for the 2D annotation of data projection.

3.4 Experiments

3.4.1 Datasets

We tested our approach on three data sets. The *Neurons* comprises 14 light microscopy scans of mouse brain, sized $250 \times 250 \times 250$. We use 10 of them for training and 4 as a validation test. *Brain* contains 13 light microscopy scans of mouse neurons, sized $216 \times 238 \times 151$. We use 10 for training and 3 for validation. *MRA* is a publicly available set of Magnetic Resonance Angiography brain scans [14]. We crop them to size $416 \times 320 \times 28$ by removing their empty margins, and use 31 annotated scans for training and 11 for validation. A sample image from each data set can be found in Fig. 3.4.

Chapter 3. Enforcing Connectivity of 3D Linear Structures Using Their 2D Projections

Table 3.1: Comparative results. A *UNet* trained with our loss function outperforms existing methods by a considerable margin in terms of the topology-aware metrics. The improvement in terms of the pixel-wise metrics is smaller but still there on average.

Dataset	Methods	Pixel-wise			Topology-aware	
		<i>Corr.</i>	<i>Compl.</i>	<i>Qual.</i>	<i>APLS</i>	<i>TLTS</i>
<i>Neurons</i>	MSE-3D	96.6	93.5	90.5	77.4	81.7
	MSE-2D	98.3	93.6	92.1	76.2	80.1
	CE	97.3	96.7	94.2	71.0	81.2
	Perc	97.6	96.7	94.5	76.6	84.1
	PHomo	97.5	96.9	94.7	81.5	83.9
	OURS-3D	98.3	96.7	95.1	87.1	89.6
	OURS-2D	97.8	96.3	94.3	91.6	87.4
<i>Brain</i>	MSE-3D	80.6	83.5	69.5	62.9	69.1
	MSE-2D	78.4	83.5	67.9	65.6	71.8
	CE	79.5	82.6	68.1	61.2	68.6
	Perc	80.1	85.0	70.2	68.9	74.5
	PHomo	81.3	84.8	71.0	69.4	75.2
	OURS-3D	79.9	86.4	70.9	75.1	80.2
	OURS-2D	80.3	85.5	70.7	76.3	81.2
<i>MRA</i>	MSE-3D	84.9	81.2	70.8	58.5	60.4
	MSE-2D	83.0	82.3	70.3	58.7	59.6
	CE	85.7	81.1	71.3	58.8	60.0
	Perc	83.4	83.9	71.9	60.9	64.5
	PHomo	85.3	83.5	72.8	62.1	65.2
	OURS-3D	81.5	89.5	74.3	70.7	72.0
	OURS-2D	80.3	87.3	71.8	70.5	71.9

3.4.2 Metrics

We use the following performance metrics. **CCQ** [100], *correctness*, *completeness*, and *quality* are similar to precision, recall, and the F1 score, but predicted foreground voxels are counted as true positives if they are closer than 3 voxels away from the ground truth ones. **APLS** [34] is defined as the mean of relative length differences between shortest paths connecting corresponding pairs of randomly selected end points in the ground truth and predicted graphs. **TLTS** [99] is the fraction of shortest paths in the prediction that are less than 15% longer or shorter than the corresponding ground truth paths.

3.4.3 Architectures and Baselines.

In all experiments, we use a 3D U-Net [81] with three max-pooling layers and two convolutional blocks. The first layer has 32 filters. Each convolution is followed by a batch-norm and dropout with a probability of 0.1. We used a batch size of 4. For data augmentation, we randomly crop volumes of size $96 \times 96 \times 96$ and flip them over the three axes. The networks were trained for 50k iterations with Adam [54], with the learning rate of $1e-3$ and weight decay of $1e-3$. At test

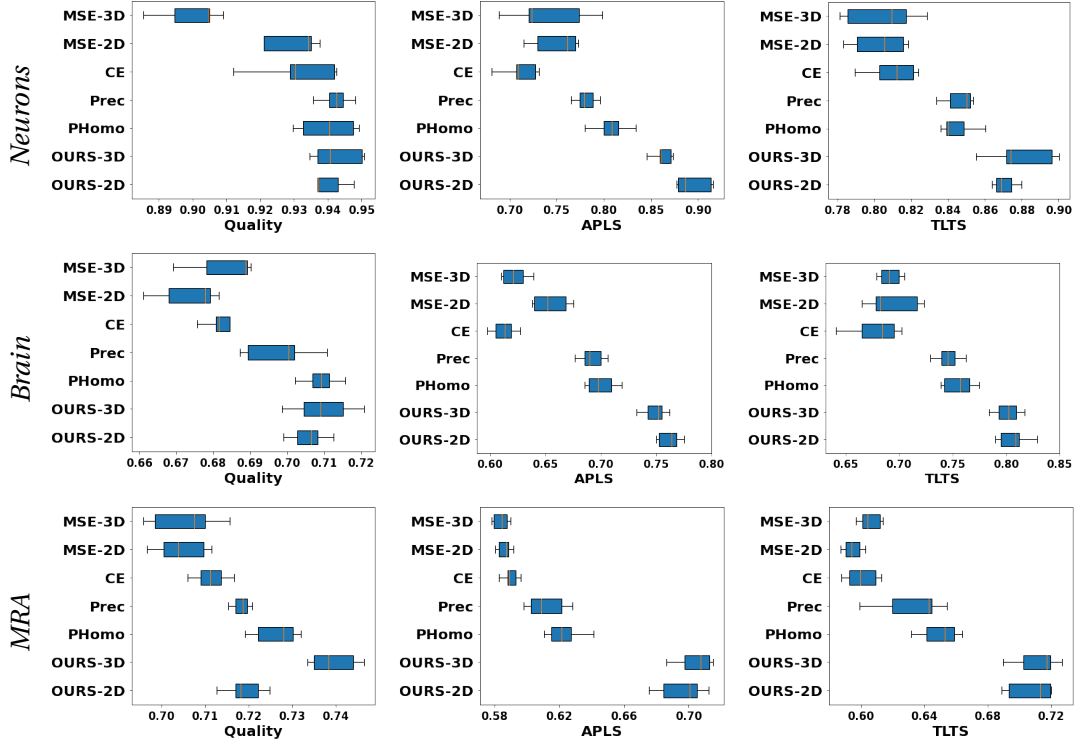


Figure 3.3: Median and quartiles over five training runs of scores attained by networks trained with different loss functions. Minimizing our topology-aware loss results in significantly higher score values than minimizing hte baselines.

time, the predicted distance map is thresholded at 2 and skeletonized to obtain centerlines. To compute the *TLTS* and *APLS*, we extract graphs from the prediction and the ground-truth, based on voxel connectivity.

As discussed in Section 3.3.3, we can train our network by minimizing either L_{3D} in Eq. 3.2 or L_{2D} in Eq. 3.3. Recall that computing L_{3D} requires 3D annotations, while 2D annotations suffice to compute L_{2D} . We will refer to these approaches as **OURS-3D** and **OURS-2D**. We compare the results we obtain in this way to:

- **MSE-3D**. L_{MSE} between 3D predictions and ground truths.
- **MSE-2D**. L_{MSE} between 2D ground truth and projected predictions [56].
- **CE**. 3D binary segmentation trained with Cross-Entropy (CE).
- **Perc**. A weighted sum of CE and a perceptual loss function that compares feature maps computed for the ground truth and predicted distance maps [71]. To extract the feature maps, we use a ResNet50 architecture pre-trained with 23 different biomedical datasets [23].
- **PHomo**. A weighted sum of CE and a loss based on Persistent Homology [48].

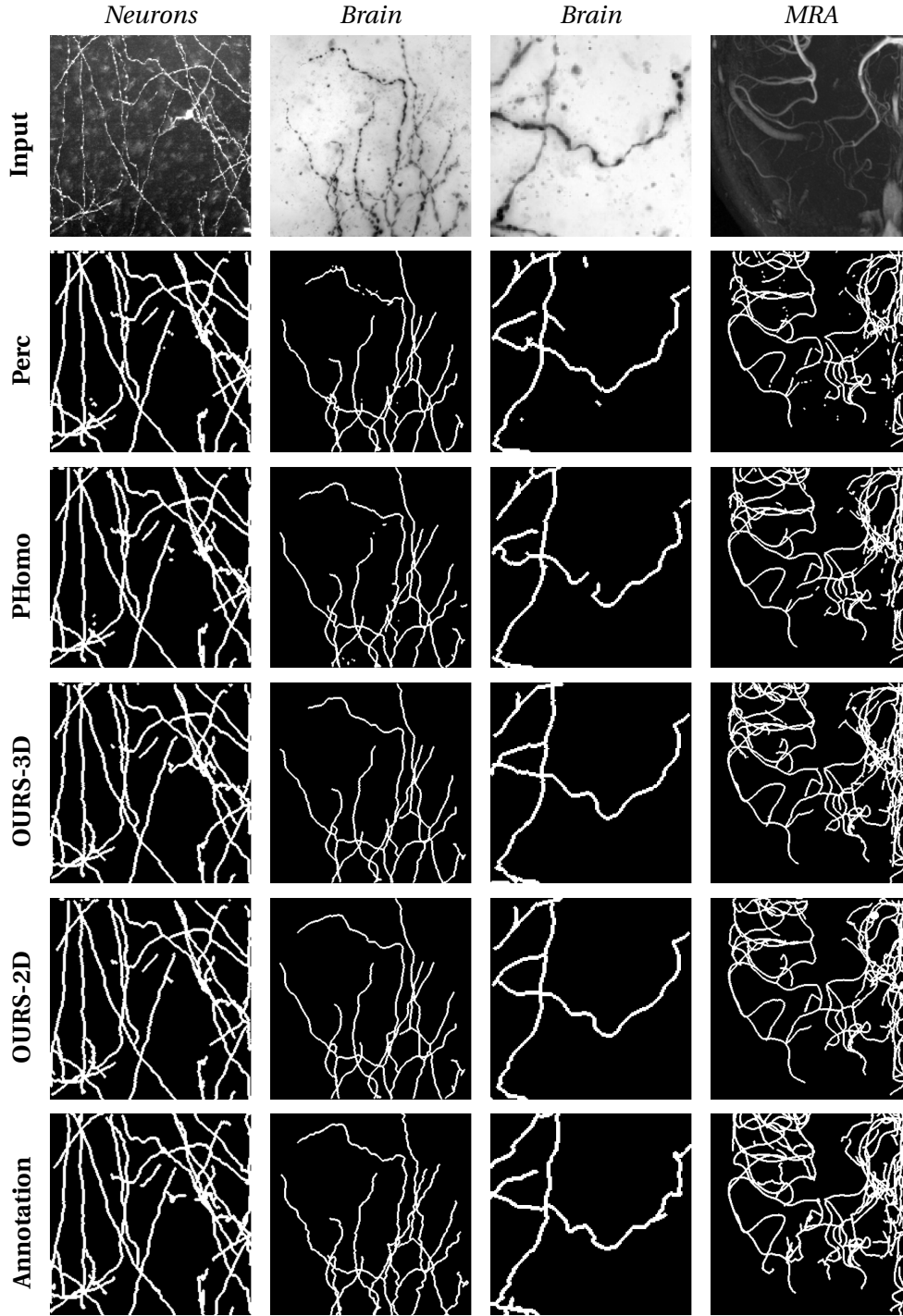


Figure 3.4: Qualitative comparison of the test results in three different datasets. The connectivity improves significantly when our approach is used.

For **Perc** and **PHomo**, we use the weighing coefficients recommended in the original publications. For **OURS-3D** and **OURS-2D**, α is set to $1e-3$ and β to 0.1. These values are selected

empirically, based on the ablation study provided in Sec. 3.4.5. We used windows of size 48 pixel to calculate L_{TOPO} .

3.4.4 Results

We provide qualitative results in Fig. 3.4 and quantitative ones in Table 3.1. Minimizing our loss function consistently improves the **APLS** and **TLTS** by a significant margin compared to minimizing pixel-wise losses. Additionally, our loss outperforms the topology-aware losses **Perc** and **PHomo**. It also delivers a boost, albeit only on average, in terms of the **CCQ**. Box plots in Fig. 3.3, show that these conclusions hold when variance of the scores is taken into account.

On average, **OURS-2D** achieves performance that are slightly lower than those of comparable performance to **OURS-3D**. However, annotating 2D slices instead of 3D stacks significantly reduces the time required to annotate, as shown in the user study conducted in [55]. Thus, when annotation effort is a concern, **OURS-2D** is an excellent alternative to **OURS-3D**.

3.4.5 Ablation Study

To investigate the impact of hyper-parameters of our method on performance, we run the following ablation studies.

Varying the Impact of the Connectivity Loss

In order to evaluate the impact of changing α on performance, we trained our *UNet* with different α on the standard split of the *Neurons* dataset. We report the results in Tab. 3.2. For low values of α , the effect of the connectivity-oriented component of the loss function is negligible. When α is increased, more and more connections are represented in the distance map.

Table 3.2: We varied α to investigate its impact on the test performance in *Neurons* dataset. The results show that setting this coefficient too-low or too-high perturbed performance, and its optimal value is in the order of $1e-3$.

α	β	Pixel-wise			Topology-aware	
		Corr.	Comp.	Qual.	APLS	TLTS
1e-2		88.2	97.0	85.9	74.7	85.3
1e-3	0.1	98.3	96.7	95.1	87.1	89.6
1e-4		98.1	96.0	94.3	81.8	87.7

Balancing Connectivity versus Dis-Connectivity

In Tab. 3.3, we report results obtained by varying β . The best scores are obtained for $\beta = 0.1$, meaning that the term preventing disconnections has ten times more impact on the loss than the term preventing false positive roads, which simply suggests that a balance between these two terms is required.

Table 3.3: We varied β to investigate its impact on the test performance in *Neurons* dataset. According to all the performance measures, the best results are obtained for $\beta = 0.1$.

α	β	Pixel-wise			Topology-aware	
		Corr.	Comp.	Qual.	APLS	TLTS
1e-3	1	96.3	96.4	92.9	85.7	87.5
	0.1	98.3	96.7	95.1	87.1	89.6
	0.01	95.9	96.9	93.1	85.3	88.9

3.5 Conclusion

We proposed a loss function that enforces topological consistency in 2D projections. Training a deep net with our loss greatly improves the 3D connectivity of its outputs and reduces the annotation effort required to obtain training data. In our current implementation, we use projection direction independently of the shape of the delineated structures. However, some projections are more informative than others. To further improve delineation accuracy while reducing the required annotation effort, we will develop algorithms for automatic selection of the optimal projection direction for different parts of the volume, so that we can use less than three projections.

4 Persistent Homology with Improved Locality Information for more Effective Delineation

Persistent Homology (PH) has been successfully used to train networks to detect curvilinear structures and to improve the topological quality of their results. However, existing methods are very global and ignore the location of topological features. In this chapter, we remedy this by introducing a new filtration function that fuses two earlier approaches: thresholding-based filtration, previously used to train deep networks to segment medical images, and filtration with height functions, typically used to compare 2D and 3D shapes. We experimentally demonstrate that deep networks trained using our PH-based loss function yield reconstructions of road networks and neuronal processes that reflect ground-truth connectivity better than networks trained with existing loss functions based on PH. This work appeared in [74].

Oner, D., Garin, A., Kozinski, M., Hess, K., and Fua, P. *Persistent Homology with Improved Locality Information for more Effective Delineation*. IEEE Transactions on Pattern Analysis and Machine Intelligence 2022.

4.1 Introduction

In many image segmentation tasks, the topology of the resulting mask is as important as, if not more than, its pixel-wise accuracy. For example, a model of an aortic valve that does not form a ring is biologically implausible. Similarly, networks of curvilinear structures—be they roads in aerial images, blood vessels in Computer Tomography (CT) scans, or dendrites and axons in Light Microscopy (LM) image stacks—should not feature breaks that disrupt connectivity or false connections between disjoint structures. Unfortunately, deep networks trained by minimizing pixel-wise loss functions, such as the cross-entropy or the mean square error, are subject to such mistakes. This is in part because it often takes very few mislabeled pixels to alter the topology significantly with little impact on the pixel-wise accuracy. In other words, it is possible for a network trained in this manner to deliver both a good pixel classification

accuracy and an incorrect topology.

Specialized solutions to this problem have been proposed in the form of loss functions that compare the topology of the prediction to that of the annotation. They are effective for specific applications but do not naturally generalize. For example, the perceptual loss of [71] penalizes topological differences between the prediction and the ground truth, but cannot be guaranteed to detect them all. Similarly, minimizing the MALIS loss for segmenting electron microscopy scans [13, 39] yields better region boundaries but does not penalize interruptions in loopy linear structures. This has been addressed by [75] for delineation of 2D road networks but the proposed solution is not applicable to 3D image stacks.

Persistent Homology (PH) [32], an elegant approach to describing and comparing topological structure of data, offers the promise to address the connectivity problem in a generic way, both for 2D and 3D images. Homology is the study of topological features in an object, such as its connected components (0-homology classes), loops (1-homology classes), and closed surfaces (2-homology classes). Persistent homology detects homology classes in objects *filtered* at different *scales*. A homology class that appears at a particular scale and disappears at a larger one is represented by a scale interval called the *persistence interval*. The set of persistence intervals for all the homology classes characterizes the overall topology of the structure. It can be represented by a *persistence diagram*. The similarity of these diagrams across two different structures can then be used to quantify their topological similarity. This has been successfully exploited to train deep networks for delineation [48], image segmentation [48, 28, 27] and crowd counting [1].

We show that these methods fail to unleash the full power of persistent homology, because they discard too much information about the structure of the prediction and the annotation when encoding them in the form of persistence diagrams. As shown in Fig. 4.1, this can result in networks that still fail to enforce the proper topology. To remedy this, we introduce a new approach to computing persistence diagrams that increases their descriptive power, as shown in Fig. 4.3. Our main contribution is a novel filtration technique that combines two approaches to filtration commonly used in topological data analysis (TDA): thresholding-based-filtration [28, 27, 48] and the height function [91]. It yields a loss function that can be used for both 2D and 3D images and significantly improves performance compared to state-of-the-art topological methods, as we will demonstrate in our experiments.

4.2 Related Work

Training a deep network that produces topologically correct segmentations has typically been done by designing loss functions that, when minimized, favor plausible topology. In this section, we briefly review first those that do not rely on Persistent Homology, and then those that do.

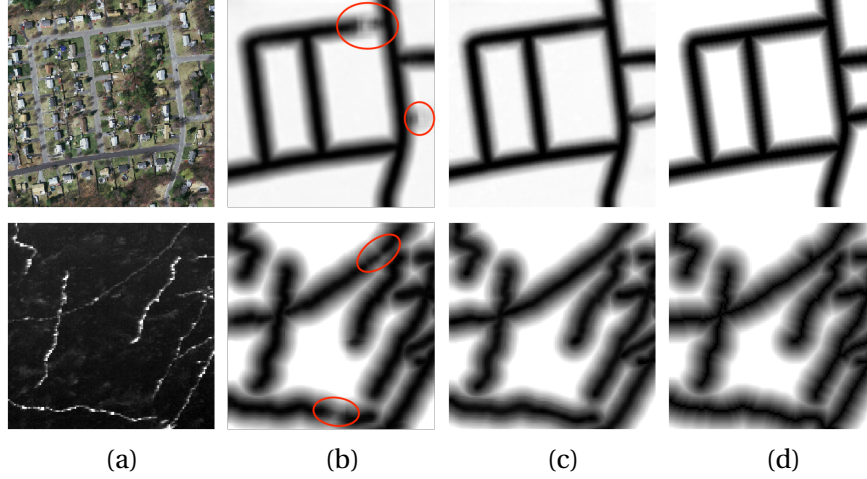


Figure 4.1: **2D and 3D delineation.** (a) Aerial image and slice of a microscopy stack. (b) A network trained using a standard homology-based loss yields road and neurite interruptions. (c) One trained using our localized loss is more topologically accurate and produces predictions that closely resemble the ground truth (d).

4.2.1 Losses Designed to Enforce Topological Correctness.

Several such losses have been proposed already to go beyond pixel-wise classification accuracy by encoding more global properties. In [61], the connectivity between neighboring pixel pairs is used as an additional source of supervision. This approach has been shown to improve connectivity, but since disconnections or false connections are not penalized explicitly, there is no guarantee it captures all such errors. The perceptual loss of [71] is based on the assumption that a pre-trained neural network can capture differences of connectivity between the prediction and the ground-truth. However, even though it has been shown experimentally to improve the topology of masks produced by a deep net, there is no guarantee that this assumption holds in general. Making the Rand index of segmentations produced by the network similar to that of ground truth ones [13, 39] helps when modeling tree-like structures, both in 2D and in 3D, but cannot prevent disconnections in loopy structures. This shortcoming has been addressed by [75] by detecting disconnections of 2D loopy structures as interconnections of background regions, but the proposed solution does not generalize to 3D.

4.2.2 Losses that rely on Persistent Homology

Persistent Homology (PH) [33, 108] is a well-established topological data descriptor. One of its important applications is comparing the topological structures of binary images, for example by enforcing the correct Betti number on binary masks resulting from inference in Markov Random Fields [22]. More recently, it has been shown that persistence diagrams can also be computed for grayscale images and differentiated with respect to the pixel values [48, 28, 40, 60, 19]. Hence, they can be incorporated into loss terms and used to train deep networks. In this vein, a loss term that enforces a sequence of desired Betti numbers on the predicted

Chapter 4. Persistent Homology with Improved Locality Information for more Effective Delineation

segmentation was introduced in [28]. This approach was further extended to a loss function that tends to equalize the Betti number of the prediction and the ground truth [27]. In a similar vein, the loss term of [48] relies on comparing persistence diagrams of the prediction and the ground truth, where the persistence diagrams are obtained by thresholding. As discussed in the next section, for binary ground truth this results in degenerate persistence diagrams that only encode the Betti number. Thus, this approach can be interpreted as equalizing the Betti numbers of the prediction and the ground truth, as in [27]. This was improved upon in [96] by applying PH to distance maps instead of binary annotations or class affinity maps. We show in the next section that this makes the loss function better at detecting and penalizing topological errors. Unfortunately, even this improved technique is susceptible to incorrectly matching the persistence diagrams of the prediction and the ground truth because it throws away location information. By incorporating such information into our diagrams, our method makes them more informative and alleviates this problem.

It has also been proposed to detect disconnections in predicted 2D and 3D structures using Discrete Morse Theory [49]. Topological features that are inconsistent with the ground truth are then penalized in the loss function. However, when the annotations lack spatial precision, which is often the case for neurite and road centerline annotation like the ones studied here, ground-truth inaccuracies may confuse the network. By contrast, our technique allows for considerable misalignment between the prediction and the ground truth.

4.3 Method

We first introduce Persistent Homology and its application to characterizing two-dimensional images and three-dimensional image stacks. As PH provides global descriptors that ignore the location of topological features, we then introduce our approach to accounting for it.

4.3.1 Persistent Homology

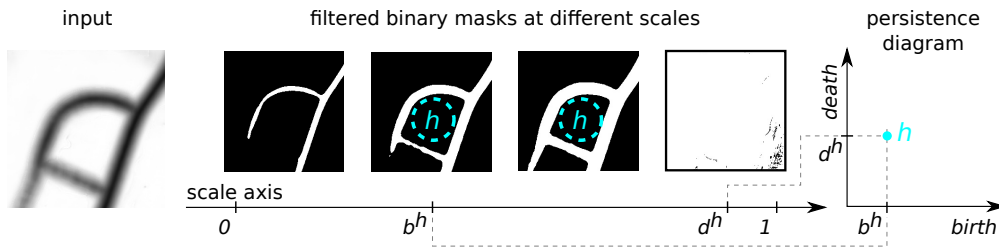


Figure 4.2: **Filtration.** When the distance map shown on the left is filtered by thresholding, the loop h emerges at scale b^h and is filled at scale d^h . This gives rise to the point (b^h, d^h) in the persistence diagram shown on the right. Here, thresholding means retaining all pixels whose value is lower than the threshold.

In the interest of simplicity, we introduce PH for binary images and image stacks, where homology classes are limited to connected components, loops, and closed surfaces. We refer the interested reader to the review [33] for a more general treatment, applicable to non-image

and higher-dimensional data.

At the heart of PH is detecting homology classes—connected components, loops, closed surfaces—at many different scales. The ones that exist over a wide scale range are called persistent and deemed more likely to represent true features, as opposed to sampling artifacts or noise. Here, scale has a very specific meaning. It refers to the parameter of a filtration function F that is applied to an image \mathbf{Y} to produce topological objects called *cubical complexes*. They arise when filtering images and their properties are described in [43, 11] for instance. A reader not familiar with algebraic topology can think of them as binary masks. The masks obtained for different scales form a sequence of inclusions, that is, for a pair of scale parameters $s_1 < s_2$, the mask $F(\mathbf{Y}, s_1)$ is entirely contained within the mask $F(\mathbf{Y}, s_2)$. The simplest example of a function for filtering grayscale images is thresholding, where the threshold acts as the scale, as shown in Fig. 4.2.

As the scale changes, homology classes in the filtered cubical complex emerge and disappear. To capture this, the scale range is sampled from small to large, the image is filtered at the selected scale values, homology classes in the resulting binary masks are detected algebraically [33], and correspondence is established between the homology classes found at consecutive scales. For each class, this yields a pair (b, d) , where b is the scale at which the homology class appears and d the scale at which it disappears. We will refer to them as *birth* and *death* times and to the interval $[b, d]$ as the *persistence interval* of the homology class. The set $P_{\mathbf{Y}} = \{(b^h, d^h)\}_{h \in H_{\mathbf{Y}}}$, where $H_{\mathbf{Y}}$ is the set of all homology classes found in the filtered image \mathbf{Y} , is called the *persistence diagram* of \mathbf{Y} , and was first introduced by [4]. In practice, we use the Gudhi library [65] to compute persistence diagrams from images. Fig. 4.2 depicts the birth and death of a specific homology class.

To compare images \mathbf{Y}_1 and \mathbf{Y}_2 , one-to-one matching is performed between their persistence diagrams, $P_{\mathbf{Y}_1}$ and $P_{\mathbf{Y}_2}$, with the cost of matching a homology $g \in H_{\mathbf{Y}_1}$ to a homology $h \in H_{\mathbf{Y}_2}$ set to $c_{g,h} = (b^h - b^g)^2 + (d^h - d^g)^2$ and the cost of leaving an interval $[b, d]$ unmatched is set to the distance between the point (b, d) and the diagonal in \mathbb{R}^2 . The optimal matching can be found using the Hungarian algorithm. Its cost that we denote as $C(\mathbf{Y}_1, \mathbf{Y}_2)$ quantifies the topological discrepancy between \mathbf{Y}_1 and \mathbf{Y}_2 by penalizing differences between corresponding homology classes and ones that only appear in either \mathbf{Y}_1 or \mathbf{Y}_2 .

4.3.2 Training Deep Networks using PH

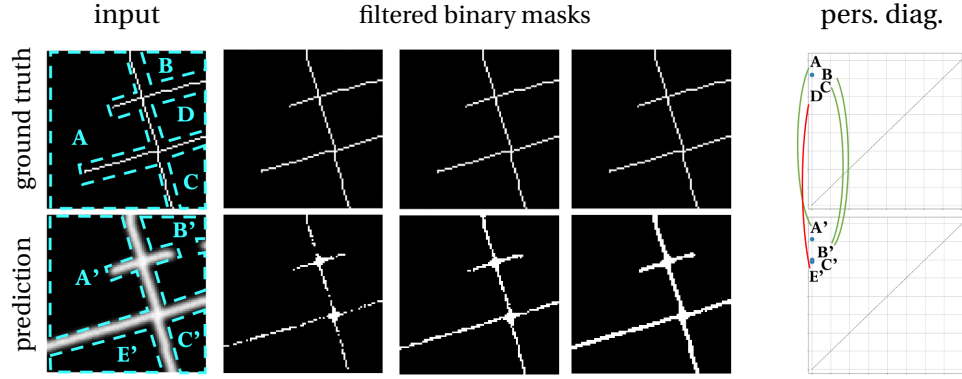
Let f be a network that associates to an input image \mathbf{X} a segmentation mask $\mathbf{Y} = f(\mathbf{X})$ such that for all pixels or voxels $p \in \mathbf{Y}$, $0 \leq \mathbf{Y}[p] \leq 1$ and let $\hat{\mathbf{Y}}$ be the corresponding ground-truth mask. A natural idea then is to train f by minimizing

$$L_{\text{tot}}(\mathbf{Y}, \hat{\mathbf{Y}}) = L(\mathbf{Y}, \hat{\mathbf{Y}}) + \alpha C(\mathbf{Y}, \hat{\mathbf{Y}}), \quad (4.1)$$

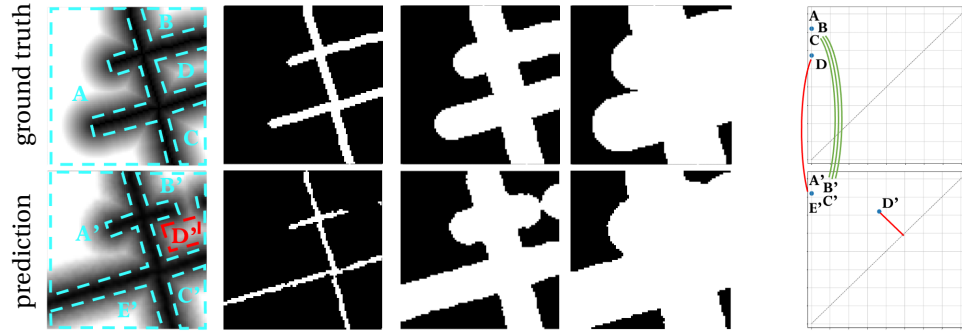
where L is the standard loss function, either the Mean Square Error, or the Cross Entropy, and α is a hyper-parameter, which we set to 0.01 in practice. This is possible because C is sub-differentiable with respect to its inputs when filtration is achieved by thresholding, as shown in [48, 28, 60]. However, when the ground truth \hat{Y} is binary, as it often is, all structures emerge at scale zero and disappear at scale one. Hence, as shown in Fig. 4.3(a) the persistence intervals all are $[0, 1]$. In other words, all points in a ground truth persistence diagram are located in its upper-left corner, and the only difference between diagrams obtained for annotations of different images is the number of points they contain. Unlike in classical applications of PH [33], where persistence diagrams serve as rich topological descriptors, such diagrams only encode the Betti number of the annotation. An approach to handling this difficulty is to replace the binary ground truth by its distance transform that can be thresholded over a wide range of threshold values to create different binary masks [96]. Unfortunately, computing the persistence diagram of a ground truth distance transform still yields persistence diagrams in which the topological features of the original, binary ground truth are spread along the ‘death’ axis but not along the ‘birth’ one: The distance value at the structures themselves is zero and, as a result, all the loops of the ground truth mask appear as soon as the scale value becomes positive. As shown in Fig. 4.3(b), this may lead to erroneous matches between persistence diagrams, which encourages the deep network to produce wrong segmentations.

4.3.3 Accounting for the Location of Topological Features during Filtration

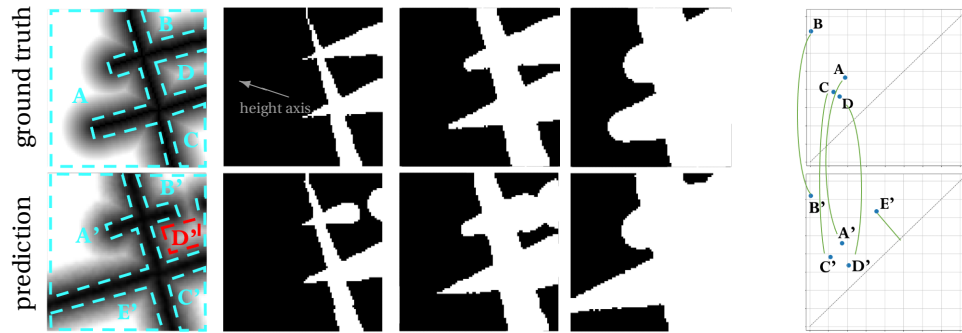
Our goal is therefore to prevent erroneous matches between topological features of the prediction and of the ground truth. To this end, we want to use the features’ image location to characterize them. However, re-defining the matching cost to include a position-dependent term would be difficult, because topological features extend across the scale-space, and because there is no natural notion of distance between them. Hence, instead of modifying the matching cost, we propose a new filtration function that distinguishes features at different positions. We draw our inspiration from a filtration technique called the height function [91]. It was originally designed for three-dimensional meshes and can be applied to binary images by assigning to each pixel a *height* value that is the coordinate of its projection along a selected straight line. Filtration is carried out by forming binary masks made of pixels whose height is smaller than the scale parameter [44]. As the scale is increased, the binary image is revealed in scan-lines perpendicular to the height axis, one scan-line at a time. The birth and death times are the heights of pixels responsible for the emergence and disappearing of homology classes. As a result, the persistence diagram contains partial information about the location of topological features. Moreover, both birth and death times of different homology classes are distributed across scales. Additionally, it has been shown that a binary image can be reconstructed from as few as four persistence diagrams obtained with height functions with well-chosen directions [9]. A height function is only defined for binary images, but the abovementioned result inspired us to extend its definition by combining it with thresholding distance maps. Given a scale s , the value of the filtered binary mask at coordinates \mathbf{p} is taken



(a) **Filtration by thresholding** binary ground truth and predicted class affinity maps. Note, that all points in the ground truth persistence diagram (*top-right*) coincide. This results in erroneous matches between the predicted and ground truth homology classes. Minimizing a loss function based on such a filtration can magnify the errors.



(b) **Filtration by thresholding distance maps** distributes the topological features of the ground truth along the vertical but not the horizontal axis. This still results in erroneous matching between the predicted and ground truth homology classes: Loop D' in the prediction emerges at a higher threshold than E', causing a faulty matching of E' to D'.



(c) **Our localized filtration of distance maps** distributes the persistence diagram of the ground truth across the plane, promoting correct matches between predicted and ground truth homology classes. The gray arrow represents the direction of the height axis used by the filtration function.

Figure 4.3: Comparing filtration functions on synthetic data. The binary ground truth road annotation (*top-left* in each table part) contains four loops, marked with cyan dashed lines. We synthesized a predicted class affinity map (*bottom-left* in each part) by extending one road to the left and interrupting another. In consequence, loop B and D from the ground truth are joined into B' in the prediction, and A is split into A' and E'. For each filtration method, we show binary masks resulting from filtration at different scales, pairs of persistence diagrams, and their optimal matches.

to be

$$F(\mathbf{Y}, s)[\mathbf{p}] = \mathbb{1}(\mathbf{Y}[\mathbf{p}] + \rho(\mathbf{p}) < s), \quad (4.2)$$

where $\mathbb{1}(\cdot)$ evaluates to one if the condition in the bracket is satisfied and to zero otherwise. In essence, this amounts to thresholding the sum of the height function ρ and the pixel values. From the perspective of TDA, such combination of two filtration functions can be seen as a line in the fibered barcode defined by [18].

In its simplest form, ρ is a linear function of pixel coordinates, and the region highlighted for any s extends along a line perpendicular to the height axis, as shown in Fig. 4.3(c). But other forms of ρ are also possible. We tested

- linear functions $\rho(\mathbf{p}) = \mathbf{w}^\top \mathbf{p}$, where \mathbf{w} is a two-vector hyper-parameter encoding the orientation of the height axis and the slope of the height function;
- a scaled distance to a point \mathbf{q} in the image, $\rho(\mathbf{p}) = a\|\mathbf{p} - \mathbf{q}\|_2$, where \mathbf{q} and a are hyper-parameters;
- the square of the height function $\rho(\mathbf{p}) = \mathbf{p}^\top \mathbf{W} \mathbf{p}$, where $\mathbf{W} = \mathbf{w}^\top \mathbf{w}$, and \mathbf{w} is the hyper parameter encoding the slope of the function and the orientation of the height axis;

The function ρ introduces localization information of the topological features into the persistence diagram. This is illustrated by Fig. 4.3 where different values of the scale parameter make homology classes appear in different parts of the image. But, because the scale parameter must be a scalar, it can only pinpoint location of topological features in 2D or 3D images along one direction. This could be addressed by evaluating the loss function many times for many different orientations of the height axis, or more generally, for many different hyper-parameters of ρ . This approach is legitimized by the theoretical result of [9] that states that four well chosen filtration directions suffice to completely represent a binary image. The problem of combining a number of different filtration functions is known in topological literature as multipersistence [17]. But current multipersistence techniques are not easily plugged into a deep learning framework for lack of results on their differentiability. Moreover, filtering the data along multiple directions would considerably slow down the training. Instead, we randomly draw the hyper-parameters of the height function at each training iteration. We show in Sec. 4.4.3 that, in practice, the simple linear function performs best.

4.4 Experiments

We first demonstrate that our loss function correlates with the number of topological errors better than standard PH-based losses. We then evaluate its performance in training deep networks to delineate road networks and neuronal arborizations.

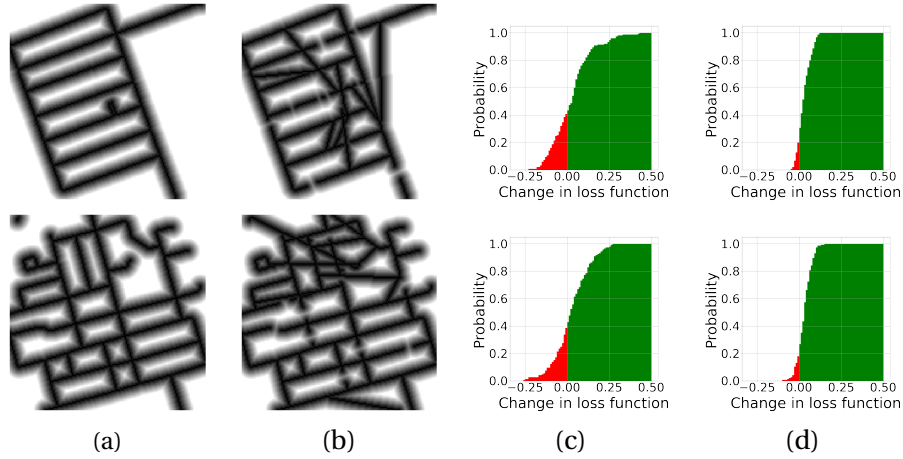


Figure 4.4: **Sensitivity of the topological loss term C to the number of injected errors** (a) Ground truth distance maps of road networks. (b) Distance maps corrupted by introducing false roads and interruptions. We randomly injected one error at a time, obtaining corrupt distance maps with 30 errors. We repeated this simulation 10 times. (c,d) The cumulative distribution function of change in the loss term in response to injecting one error. In (c), C is evaluated using the filtration by thresholding distance maps, whereas in (d) we use our filtration. The probability of decreasing the existing loss term by injecting additional errors is around 0.4, whereas for our loss term it drops to 0.2. We conclude that our loss term is more monotonic with respect to the error number.

4.4.1 Correlation to the Number of Topological Errors

We motivated our filtration technique by the fact that it introduces partial localization of topological features into the persistence diagrams and better spreads the diagrams across the plane. Here, we validate this on synthetic data to show that it correlates better with the number of errors injected into a distance map than the baseline loss, which is based solely on thresholding distance maps. To this end, we took two crops of ground truth road graphs of the *RTracer* dataset [6] and generated faulty synthetic distance maps by injecting thirty errors one at a time. They were selected randomly and with equal probability between a road disconnection and a false interconnection. After each error injection, we evaluated the topological loss term C of Eq. 4.1 using either filtration by thresholding distance maps or our combined filtration. Ideally, we would expect C to increase every time an error is added. Hence, we repeated the experiment ten times. For each crop, we plot the distribution of the increment in C resulting from adding one error, when using the baseline loss in Fig. 4.4(c) and ours in Fig. 4.4(d). The parts of the distributions shown in green correspond to positive increments, which are what we expect, and those in red denote the negative ones, which are essentially erroneous. Note that the red parts are far smaller when using our loss than the baseline one.

4.4.2 Performance in Training Deep Networks

Having shown that our loss function captures topological correctness better than existing PH-based methods, we now compare the performance of deep nets trained with our and

Chapter 4. Persistent Homology with Improved Locality Information for more Effective Delineation

existing losses.

Datasets

We experimented on four datasets.

- *RTracer*. A dataset of high-resolution satellite images covering urban areas of forty cities in six countries [6]. The ground truth was obtained from OpenStreetMap. Like [6, 62, 103, 70], we used twenty five cities as the training set and the remaining fifteen as the test set.
- *Massachusetts*. The Massachusetts dataset [68] features both urban and rural neighborhoods, with many different kinds of roads ranging from small paths to highways. For a fair comparison to [48], we split the data into three equal folds and performed a three-way cross validation.
- *Neurons*. The dataset is a part of a proprietary 3D, 2-photon microscopy scan of a whole mouse brain. It contains 14 stacks of size $250 \times 250 \times 200$ voxels and a spatial resolution of $1.0 \times 0.3 \times 0.3 \mu m$. We used ten stacks for training and the remaining four for testing.
- *Brain*. The dataset contains two 3D images of neurons in a mouse brain. The axons and dendrites have been outlined manually while viewing the sample under a microscope and the image has been captured later. The sample deformed in the meantime, resulting in a misalignment between the annotation and the image. To ensure that the test and training data comes from the same distribution, we split the two scans into stacks of $150 \times 200 \times 200$ voxels and a spatial resolution of $1 \mu m$, and randomly divided the resulting data set into a training set of twelve stacks and a test set of ten scans.

Baselines

To test the impact of our proposed filtration functions, we used the standard U-Net architecture [81], with four blocks, each with two sequences of convolution-ReLU-batch normalization. Max-pooling in 2×2 windows followed each of the blocks. The initial feature size was set to 32 and grew to 512 in the smallest feature map in the network. We augmented the training data with vertical and horizontal flips and random rotations, and used the ADAM algorithm [54] with the learning rate set to $1e-4$. We then used different version of the L_{tot} of Eq. 4.1 we minimized to train the network. We tested the following as baselines:

- *UNet-CE*. L is the Cross Entropy loss for pixel classification and there is no topological discrepancy loss, that is, $\alpha = 0.0$. Binary masks are used as ground truth.
- *UNet-MSE*. L is the mean squared error of the truncated distance to the closest foreground pixel, with no topological discrepancy loss.

- *Homo-Pre*. L is the cross Entropy loss and we compute C by thresholding pixel classification maps, as in [48, 28, 27].
- *Homo-Reg*. L is the mean squared error and we compute C by thresholding the truncated distance maps, as in [96].
- *Ours*. L is the mean squared error and we compute C using our proposed filtration function.

Based on the results of the ablation study on the *Massachusetts* data set, presented in Sec. 4.4.3, in all our experiments with *Homo-Pre*, *Homo-Reg*, and *Ours*, we set $\alpha = 0.01$ and compute the loss in windows of size 64×64 pixels. Like [48], we limit the method to homology classes order 1, that is, loops. This has two advantages. First, by convention, loops are created by the borders of the window, making disconnections in dead-ending roads or neurites detected as broken loops. Second, detection of homology classes is computationally expensive, and the time grows cubically with the number of pixels. In our current setup, computing the loss for a single window takes 0.5 seconds. Similarly to [48], we did not observe any performance gain due to using homology classes of order 0—connected components—in addition to loops.

For completeness, we also compared our approach to recent techniques *not* relying on persistent homology: *Segmentation* [6], *RoadTracer* [6], *Seg-Path* [70], *RCNNU-Net* [103], *DeepRoad* [66], *PolyMapper* [62], *DMT* [49], and *ConnLoss* [75]. *Segmentation*, *RoadTracer*, *RCNNU-Net*, and *PolyMapper* do not explicitly enforce topology constraints, while the others do and are discussed in the related work section. The outputs of these methods were shared by the authors directly with us or on the Internet, and we computed all the performance metrics.

Performance metrics

Comparing connectivity of segmentation masks is difficult, because the reconstructions rarely overlap with the ground truth, and often deviate from it significantly. There seems to be no consensus concerning the best evaluation technique; we found five connectivity-oriented metrics in concurrently published recent work. To provide an exhaustive evaluation, we used all of them.

- *APLS*. Average Path Length Similarity aggregates relative length differences of shortest paths between pairs of corresponding points in the ground truth and predicted maps [34].
- *TLTS* is a statistics of lengths of shortest paths between corresponding pairs of end points randomly selected in the predicted and ground-truth networks [99]. We report the fraction of paths with relative length difference within 5%.
- *JCT*. It is a junction score that considers the number of roads intersecting at each junction [6]. It consists of road recall, averaged over the intersections of the ground-truth and road precision, averaged over the intersections of the prediction. We report

Chapter 4. Persistent Homology with Improved Locality Information for more Effective Delineation

the corresponding F1 score.

- *Betti*. The Betti error [48] is an average absolute difference between the number of topological structures seen in the ground truth and predicted delineations. We take random patches sized 64×64 from predictions, compute the number of 1-homology classes (loops) and compare the numbers computed for the prediction and the ground truth. We average this difference over 10 trials. In practice, to compute the error we use the code made publicly available by the authors.
- *CCQ* We complement the connectivity-oriented metrics with the most popular metric that measures spatial co-occurrence of annotated and predicted road pixels. The Correctness, Completeness and Quality are equivalent to precision, recall and intersection-over-union, with the definition of a true positive relaxed from spatial coincidence of prediction and annotation to co-occurrence within a distance of 5 pixels [100]. We report the Quality as our single-number metric.

Table 4.1: Validation results on the *Massachusetts* dataset. Our loss function outperforms all PH-based loss functions. We report means and standard deviations over three independent training runs.

Method	Connectivity-oriented				pixel-based
	<i>APLS</i> \uparrow	<i>TLTS</i> \uparrow	<i>JCT</i> \uparrow	<i>Betti</i> \downarrow	<i>CCQ</i> \uparrow
<i>UNet-CE</i>	60.9 ± 3.9	41.6 ± 4.1	72.0 ± 2.7	3.12 ± 0.6	66.9 ± 2.6
<i>UNet-MSE</i>	61.3 ± 3.7	41.9 ± 4.2	71.9 ± 2.9	3.09 ± 0.7	67.3 ± 2.3
<i>DMT</i>	64.7 ± 2.9	45.8 ± 2.8	80.6 ± 2.4	0.99 ± 0.4	74.9 ± 1.9
<i>ConnLoss</i>	73.4 ± 3.6	53.2 ± 4.4	81.4 ± 1.9	1.29 ± 0.5	75.8 ± 2.2
<i>Homo-Pre</i>	62.5 ± 1.9	42.1 ± 1.9	74.2 ± 1.7	1.28 ± 0.3	69.3 ± 1.9
<i>Homo-Reg</i>	65.0 ± 2.2	45.6 ± 1.8	76.9 ± 1.9	1.09 ± 0.2	71.8 ± 2.1
<i>Ours</i>	68.7 ± 1.2	50.6 ± 2.3	79.2 ± 2.6	0.90 ± 0.3	74.9 ± 1.8

Comparative Results

We present validation results for the *Massachusetts* data set in Tab. 4.1, and test results for the *RTracer* data set in Tab. 4.2. Our method outperforms the other methods based on Persistent Homology, which demonstrates that our approach to filtering is truly effective. It also outperforms the other 2D tracing algorithms targeted at handling aerial images, *RoadTracer*, *Seg-Path*, *DeepRoad*, and *PolyMapper*, with the exception of *ConnLoss* that does marginally better. This is presumably because *ConnLoss* explicitly penalizes each disconnection of the prediction, whereas a persistence diagram is a lossy topological descriptor that may fail to penalize some errors. However, *ConnLoss* does not naturally extend to 3D data, whereas our method does. On the 3D *Neurons* data set, it outperforms the competing algorithms, as evidenced by the test results shown in Tabs 4.3 and 4.4. The qualitative results for the *Massachusetts* data set can be found in Fig. 4.5, the corresponding results for the *RTracer* data set in Fig. 4.6, and for

Table 4.2: Our loss function outperforms all PH-based loss functions on the *RTracer* dataset. We report means and standard deviations over cities from the test set.

Method	Connectivity-oriented				pixel-based
	<i>APLS</i> \uparrow	<i>TLTS</i> \uparrow	<i>JCT</i> \uparrow	<i>Betti</i> \downarrow	<i>CCQ</i> \uparrow
<i>UNet-CE</i>	63.4 \pm 1.6	37.5 \pm 1.9	78.0 \pm 1.0	3.08 \pm 0.6	59.7 \pm 2.2
<i>UNet-MSE</i>	66.3 \pm 1.9	40.0 \pm 2.0	77.5 \pm 1.3	2.99 \pm 0.5	59.5 \pm 1.9
<i>Segmentation</i>	62.5 \pm 1.5	33.0 \pm 1.6	78.2 \pm 1.5	3.04 \pm 0.6	54.4 \pm 1.0
<i>RoadTracer</i>	59.1 \pm 0.8	40.6 \pm 1.5	81.2 \pm 1.6	2.85 \pm 0.7	47.8 \pm 1.6
<i>Seg-Path</i>	68.1 \pm 1.4	46.5 \pm 1.7	75.4 \pm 1.3	2.31 \pm 0.4	54.0 \pm 1.4
<i>RCNNU-Net</i>	48.2 \pm 1.6	18.4 \pm 1.9	75.9 \pm 1.4	3.25 \pm 0.7	62.8 \pm 1.5
<i>DeepRoad</i>	24.6 \pm 2.2	6.4 \pm 0.9	51.4 \pm 1.5	4.95 \pm 1.0	43.6 \pm 2.0
<i>PolyMapper</i>	61.3 \pm 2.3	31.5 \pm 1.9	80.0 \pm 1.2	2.90 \pm 0.4	35.7 \pm 1.4
<i>ConnLoss</i>	75.4 \pm 1.6	49.6 \pm 1.4	82.6 \pm 0.6	1.30 \pm 0.4	68.4 \pm 0.9
<i>Homo-Pre</i>	67.3 \pm 1.7	42.3 \pm 1.1	78.7 \pm 0.9	1.32 \pm 0.3	61.9 \pm 1.9
<i>Homo-Reg</i>	69.9 \pm 1.6	45.1 \pm 1.4	79.6 \pm 1.3	1.07 \pm 0.3	63.2 \pm 1.6
<i>Ours</i>	73.8 \pm 1.8	47.8 \pm 0.9	81.3 \pm 1.6	0.89 \pm 0.2	66.3 \pm 1.7

Table 4.3: Comparative results on the *Neurons* dataset. Our loss outperforms all the baselines. We report means and standard deviations over three independent training runs.

Method	Connectivity-oriented			pixel-based
	<i>APLS</i> \uparrow	<i>TLTS</i> \uparrow	<i>Betti</i> \downarrow	<i>CCQ</i> \uparrow
<i>UNet-CE</i>	79.9 \pm 1.5	80.8 \pm 2.2	2.33 \pm 0.6	90.6 \pm 2.0
<i>UNet-MSE</i>	80.2 \pm 1.6	80.9 \pm 2.0	2.31 \pm 0.7	90.4 \pm 1.9
<i>Homo-Pre</i>	83.5 \pm 1.0	82.1 \pm 1.7	1.06 \pm 0.2	91.2 \pm 1.8
<i>Homo-Reg</i>	85.4 \pm 1.2	83.4 \pm 1.5	0.91 \pm 0.2	92.5 \pm 1.6
<i>Ours</i>	86.9 \pm 1.1	85.2 \pm 1.9	0.80 \pm 0.2	93.3 \pm 1.9

Table 4.4: Comparative results on the *Brain* dataset. Our loss outperforms all PH-based losses. Means and standard deviations over three independent training runs as presented.

Method	Connectivity-oriented			pixel-based
	<i>APLS</i> \uparrow	<i>TLTS</i> \uparrow	<i>Betti</i> \downarrow	<i>CCQ</i> \uparrow
<i>UNet-CE</i>	65.8 \pm 1.8	63.6 \pm 1.3	2.89 \pm 0.4	70.4 \pm 1.9
<i>UNet-MSE</i>	66.0 \pm 1.6	63.9 \pm 1.4	2.92 \pm 0.5	70.6 \pm 1.8
<i>Homo-Pre</i>	67.6 \pm 1.5	65.3 \pm 1.0	1.39 \pm 0.2	71.5 \pm 1.4
<i>Homo-Reg</i>	70.5 \pm 1.5	68.8 \pm 0.9	1.22 \pm 0.3	72.6 \pm 1.3
<i>Ours</i>	73.4 \pm 1.4	70.1 \pm 1.1	1.06 \pm 0.2	73.2 \pm 1.2

the *Neurons* data set in Fig. 4.7. For each method, we display the thresholded predictions with their skeletons overlaid in red. In the case of the 3D dataset, the images we show are maximum intensity projections.

4.4.3 Ablation Study

To investigate the impact of hyper-parameter choices on performance, we ran three ablation studies.

Weighting the PH Loss

We varied the coefficient α in Eq. 4.1, while keeping the other parameters fixed. We report the results in Tab. 4.5. The best results are achieved for $\alpha = 0.01$, and the performance decreases when α is set ten times higher or lower. This suggests that the standard Mean Square Loss is still important for overall performance, which is not a surprise, as the gradient of our

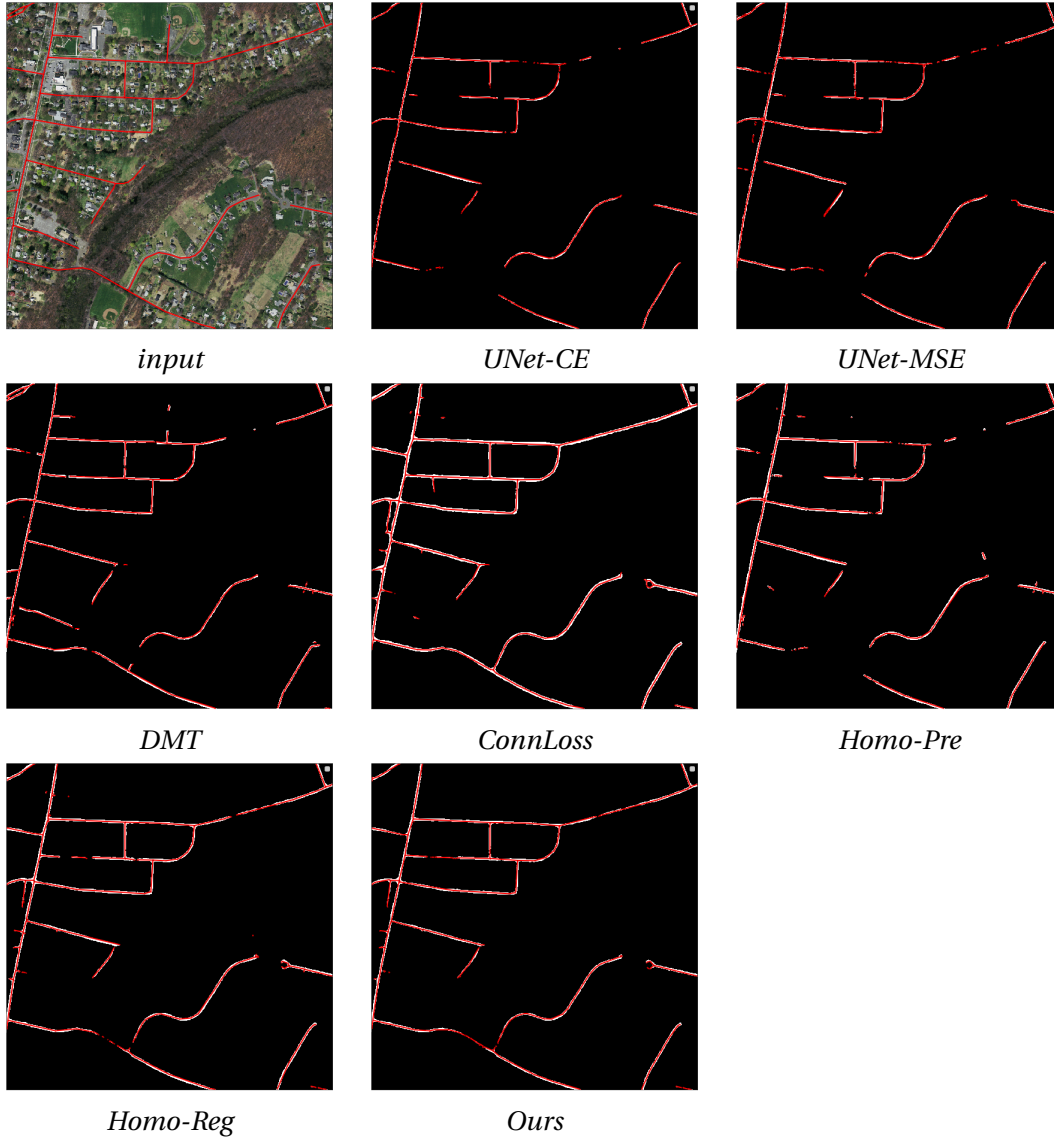
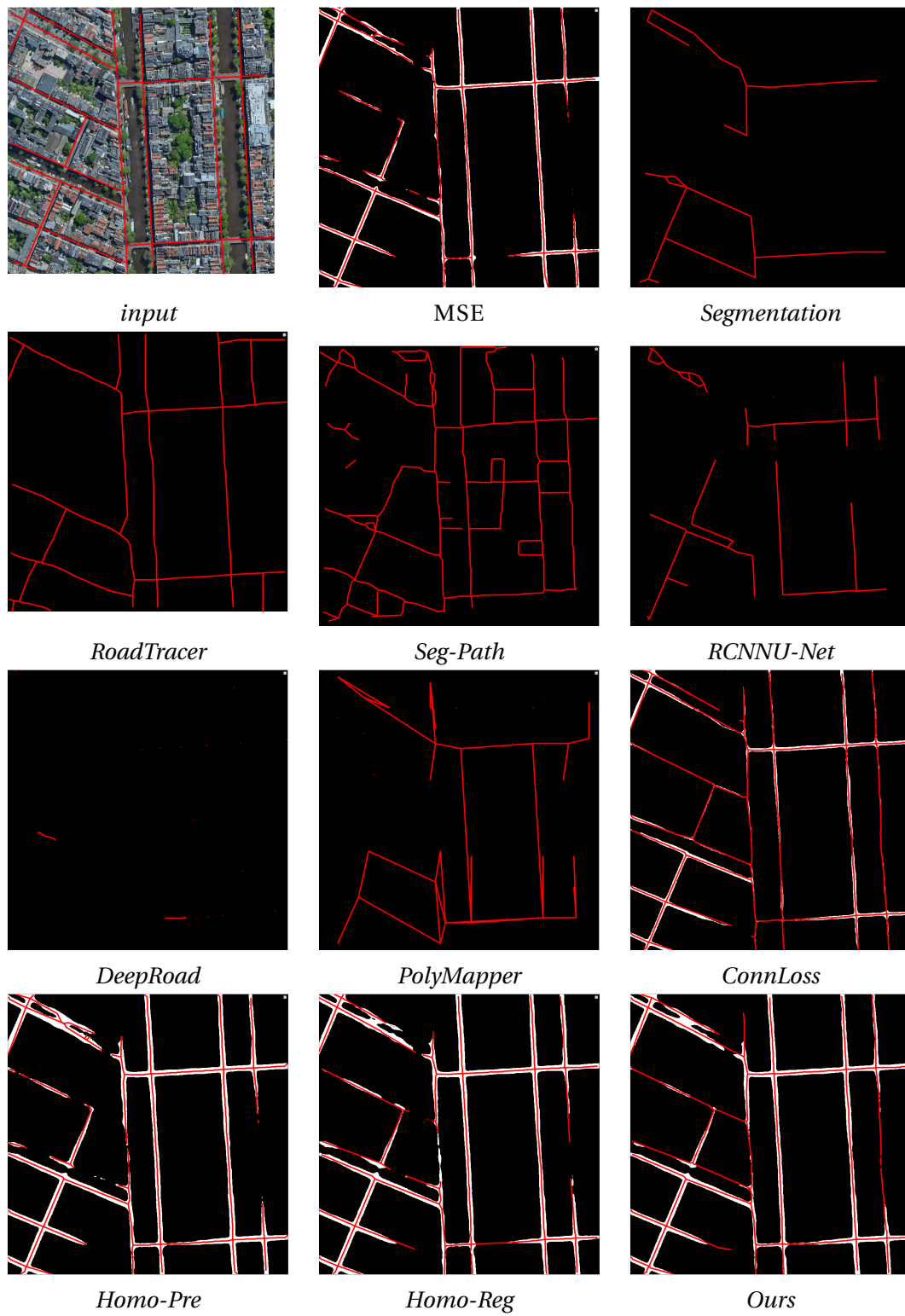


Figure 4.5: Qualitative results on the *Massachusetts* dataset.

Figure 4.6: Comparative results on the *RTracer* dataset.

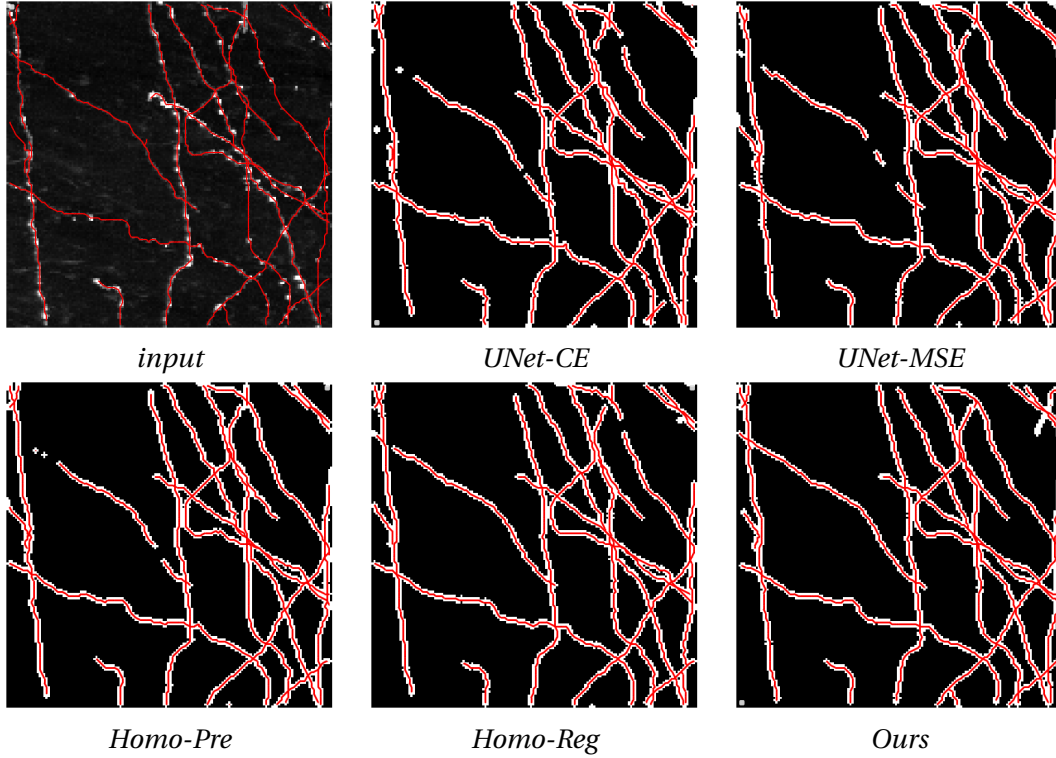


Figure 4.7: Comparative results on the 3D *Neurons* dataset.

Table 4.5: Impact of changing the learning coefficient of localized PH loss on the *Massachusetts* dataset. The window size is fixed to 64x64.

α	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>Betti</i>	<i>CCQ</i>
1e-3	64.9	46.0	77.1	1.21	72.3
1e-2	68.7	50.6	79.2	0.90	74.9
1e-1	67.1	48.9	77.8	0.94	74.6
1e-0	64.8	45.8	76.2	1.10	72.0

persistent-homology-based loss is sparse and concentrated at pixels critical for topological correctness.

Window size

We changed the size of the window in which the persistent homology is computed. We report the results in Tab. 4.6. Our method performs best when using large windows that contain significant portions of the structures of interest. We could not try even larger ones because it would have increased the time needed to detect the homologies and slowed down the training

Table 4.6: Impact of changing the window size when computing our localized loss on the *Massachusetts* dataset. The learning coefficient is fixed to $1e-2$.

Window Size	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>Betti</i>	<i>CCQ</i>
8x8	62.1	41.9	73.0	2.84	67.2
16x16	62.7	42.4	74.5	2.09	68.8
32x32	65.4	45.7	77.1	1.17	72.5
64x64	68.7	50.6	79.2	0.90	74.9

too much.

Table 4.7: Performances of different height functions used for localized PH loss on the *Massachusetts* dataset. The learning coefficient is fixed to $1e-2$ and window size to 64x64

Height Function	Connectivity-oriented				pixel-based
	<i>APLS</i>	<i>TLTS</i>	<i>JCT</i>	<i>Betti</i>	<i>CCQ</i>
Dist. to a point	67.8	49.4	77.9	1.01	73.6
Random Linear	68.7	50.6	79.2	0.90	74.9
Fixed Linear	67.5	48.7	76.5	1.15	73.0
Square	64.2	45.1	76.3	1.32	70.3

Height Functions

We also evaluated the effect on performance of using different forms of function ρ in Eq. 4.2, that ties homology birth and death times to image coordinates, distributing the points in the persistence diagram. We present the results in Tab. 4.7. The distance to a random image point, or the use of a quadratic instead of linear function of image coordinates do not result in higher performance than the plain linear function.

4.5 Conclusion

We demonstrated a fault in the design of existing methods to employ Persistent Homology to train deep networks in delineating curvilinear structures: by using inadequate filtration functions, they severely reduce the information content of the persistence diagrams, hampering performance of the trained network. We proposed an improved approach, based on combining filtration by thresholding with the height function, that increases the descriptive power of the diagrams, and gives PH a place among the best-performing methods to train topologically accurate deep networks.

Chapter 4. Persistent Homology with Improved Locality Information for more Effective Delineation

The proposed approach is limited by the need to randomly select the parameters of the height function at each training iteration, because some orientations of the height axis might result in a failure to detect topological errors, or provoke erroneous matches between the persistence diagrams of the prediction and the ground truth. We therefore plan to investigate the use of multipersistence [17] for less random and more effective supervision.

Another limitation stems from the fact that our loss function has sparse gradients that only depend on values at pixels that are critical for emergence and disappearance of topological features. This limits robustness and our future work will focus on developing topological descriptors with more smooth gradients.

While our loss function improves the topological correctness of the segmentation masks, some bio-medical applications require full confidence of correctness of anatomy models, which current methods cannot guarantee. This also motivates us to investigate the use of topological methods to highlight the regions of the segmentation masks that require manual correction, thereby facilitating proof-reading of segmentation results.

5 Adjusting The Ground Truth Annotations for Connectivity-Based Learning to Delineate

Deep learning-based approaches to delineating 3D structure depend on accurate annotations to train the networks. Yet in practice, people, no matter how conscientious, have trouble precisely delineating in 3D and on a large scale, in part because the data is often hard to interpret visually and in part because the 3D interfaces are awkward to use.

We introduce a method that explicitly accounts for annotation inaccuracies. To this end, we treat the annotations as active contour models that can deform themselves while preserving their topology. This enables us to jointly train the network and correct potential errors in the original annotations. The result is an approach that boosts performance of deep networks trained with potentially inaccurate annotations. This work appeared in [73].

Oner, D., Citraro, L., Kozinski, M., and Fua, P. *Adjusting the Ground Truth Annotations for Connectivity-Based Learning to Delineate*. In IEEE Transactions on Medical Imaging 2022.

5.1 Introduction

As in many areas of computer vision, deep networks now deliver state-of-the-art results for delineation tasks, such as finding axons and dendrites in 3D light microscopy images. However, their performance depends critically on the accuracy of the ground-truth data used to train them. This is especially true when the delineation task is treated as a segmentation one and the network is trained by minimizing the cross-entropy between the centerline predictions and ground-truth annotations, which is one of the most popular paradigms.

In practice, these so-called ground-truth annotations are usually supplied manually by an annotator who may not draw with the utmost accuracy and can therefore easily be a few voxels off the true centerline. This is not a matter of carelessness but a consequence of 3D delineation being truly difficult to do well on a large scale. As a result, inaccurate annotations are more the

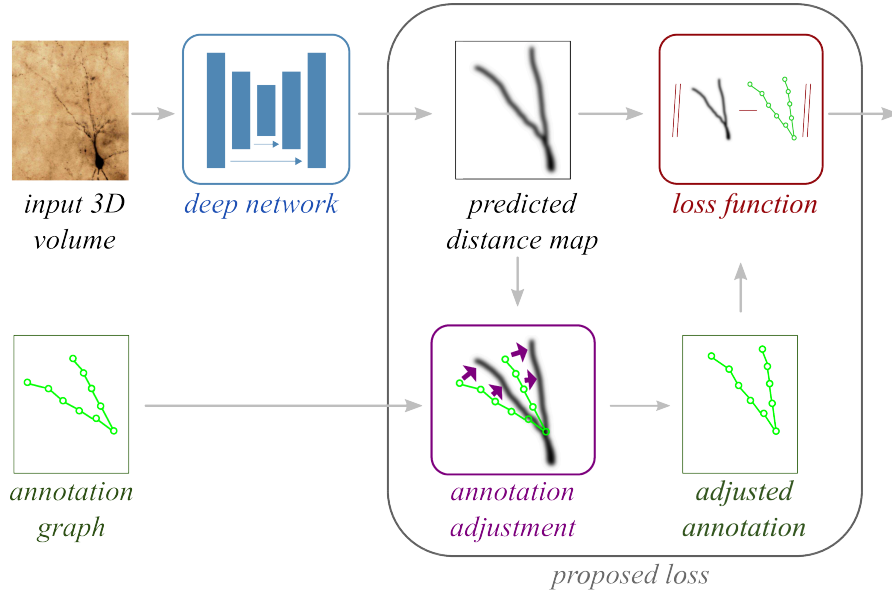


Figure 5.1: **Our approach.** To account for annotation inaccuracies during training, we jointly train the network and adjust the annotations while preserving their topology.

rule than the exception and this adversely affects how well the networks ultimately perform. One solution would be to have several annotators delineate the same data and combine their delineations. However, this would turn an already tedious, slow, and expensive process into an even slower and more expensive one that almost no one can afford.

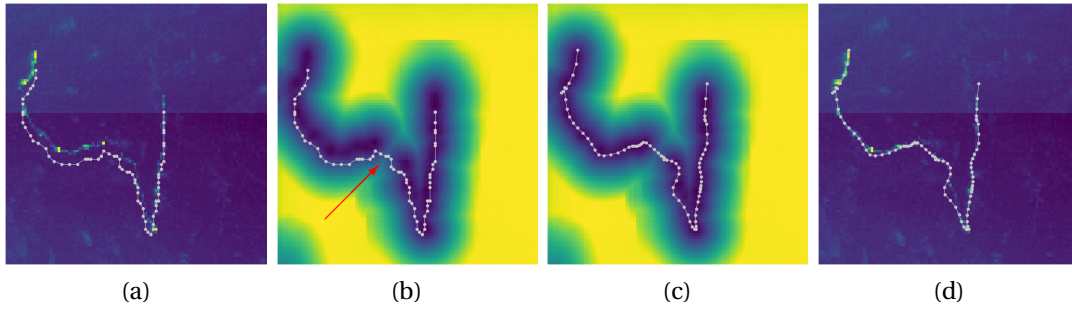


Figure 5.2: **Correcting an inaccurate annotation.** (a) A microscopy scan of a neurite with an inaccurate annotation overlaid in white. (b) Distance map predicted by the deep net. Ideally, the pixels crossed by the centerline should have value zero (dark color). In practice, this is not always the case. There are non-zero values in the area indicated by the red arrow, presumably because the neurite is hardly visible there. Nevertheless, the distance map is sufficiently good to adjust the annotation. The adjusted annotation is shown in (c) and (d). This network retrained with adjusted annotations can now generate a better distance map even where the neurite is barely visible.

We introduce a method that explicitly accounts for annotation inaccuracies and delivers the same performance as if they were perfectly accurate. Our main insight is that the annotations are usually imprecise more in terms of the 3D location of the centerlines than of the topology of the graph they define. We can therefore treat them as deformable contours forming a graph

that can be refined by moving its nodes while preserving its structure. We cast this approach to training a deep network as a joint optimization over the network parameters and node positions. We then show that we can eliminate the node variables from the optimization problem, which can then be solved by minimizing a loss function. This loss function accounts for the annotation's lack of spatial precision. It can be minimized in the traditional manner and the output of the re-trained network used to refine the annotation.

Fig. 5.1 depicts our approach and Fig. 5.2 showcases its behavior. We will demonstrate that it brings substantial improvements when training networks to delineate neurons in two-photon and confocal microscopy image stacks. Hence, our contribution is an automated approach to better leveraging inaccurate training data, which, in our experience, represents the vast majority of data available to practitioners.

5.2 Related Work

5.2.1 Automated Delineation

Automatic delineation of curvilinear structures has been an active research topic for decades. It has evolved from manually designing filters that respond strongly to tubular structures [36, 58, 89] to feeding hand-designed features into boosted trees [102, 12], support vector machines [50], or GradientBoost [86], and finally to fully relying on neural networks [69, 41, 63, 78, 70].

The latter now routinely deliver the best performance when properly trained. However, obtaining accurately annotated data, especially in 3D, is a challenge. In practice it is rarely available in sufficient quantities. And what annotated data there is, is rarely accurate because manually delineating 3D structures is challenging. Introducing a degree of self-supervision is a way to address this difficulty [8, 31] but this does not detract from the fact that the training would work even better if the available annotated data were accurate. This can be partially ascribed to the fact that most current networks are trained by minimizing the standard cross entropy or differentiable intersection-over-union loss [66]. As pixel-wise measures, both are sensitive to even small misplacements of the linear structures' centerlines. In [71], this is partially addressed by introducing a loss component that accounts for global statistics of the network output, but the cross entropy remains a key component of the overall loss. Similarly, the method of [75] relies on introducing a topology-preserving term but still depends on the annotation being accurate.

Accuracy can be improved by having several people annotate and combining their results using robust statistics. This is effective but even more expensive than obtaining one set of annotations and therefore out of reach for most practitioners. The problem can be partially alleviated by annotating only in 2D projections of the 3D data volumes [77, 106, 56], which is easier, but may result in even less precise annotations than those performed in 3D.

Chapter 5. Adjusting The Ground Truth Annotations for Connectivity-Based Learning to Delineate

A similar problem to the one we address here also arises in the context of two-dimensional semantic boundary detection. The outlines one finds in annotated training sets are often rather imprecise and training the networks to nevertheless discover contours that overlap with them is an issue. In [104], training is reformulated as simultaneously optimizing the parameters of a deep net and correcting the annotations by solving a mixed binary-continuous optimization problem. However, unlike in our approach, preservation of annotation topology is not warranted and the corrections may break the continuity of annotations. This is a major problem when tracing neurons or blood vessels, because topology changes influence the biological interpretation of the results. The same problem is addressed in [2] by proposing a neural layer and a loss function that can be added on top of an edge detector and make it possible to find more accurate contours than those in the annotations. However, because the regions are represented in an implicit fashion, there is no more guarantee than in [104] that the annotations' connectivity will be preserved. Connectivity being at the heart of our applications, we therefore chose to use explicit deformable models, such as those described below.

5.2.2 Handling Noisy Annotations

Even though we know of no other algorithm that adjusts the geometry of centerline annotations during training, explicitly accounting for the fact that the annotations are noisy has received some attention. In [97], annotations produced by non-expert annotators are accommodated by means of a dedicated distillation architecture and a noise-robust Dice loss. In [20], a dedicated network architecture and a semi-supervised training routine encourage equivariance to deformations to handle potential inaccuracies resulting from using a heuristic annotation tool. In [107], annotation noise is handled by a quality assessment module that discounts the loss in regions where the estimated label quality is low. Similarly, in [67], a distillation training setup and architecture based on self-attention are used to suppress the influence of erroneous labels on the trained network. In contrast to all these approaches, ours explicitly distinguishes between inaccuracies in position and topological errors. Because the former occur far more frequently than the latter, our loss function adjusts the centerline locations, while preserving the topology of the annotations.

5.2.3 Deformable Contour Models

Deformable contours [53, 88, 38] were initially introduced as a means to semi-automatically delineate simple contours while imposing smoothness constraints on the resulting outlines. They were later generalized to model network structures [37, 15] that can deform while preserving their topology. They are therefore well suited for refining our inaccurate annotations under the assumption they are topologically correct but that their locations are imprecise.

More recent deformable contours rely on minimizing energy functions generated by deep networks [64, 24, 97, 47], which enables end-to-end learning. Unlike in these methods, which

rely on evolving the contour for segmenting the image at test time, our use of deformable contours is limited to adjusting the annotations during training.

Active appearance models [29] enable modelling the appearance of imaged objects, in addition to their shape. They can be learnt from coarse annotations, which are adjusted when fitting the model to the data [80]. The level of detail of the active appearance model can then be increased and, before the more detailed model is fitted to the data, it can be initialized with the parameters of its less detailed version. In this work, we also adjust the annotation during learning, but represent them as network snakes, and train a deep convolutional network, instead of fitting an active appearance model.

5.3 Method

Given a set of microscopy stacks along with the corresponding and possibly imprecise center-line annotations, we want to train a deep net to produce precise delineation. To this end, when training the deep network, we adjust not only its weights but also the annotations themselves. We first present the vanilla training procedure without annotation adjustment and explain why it is sub-optimal when the annotations lack precision. We then formalize our training procedure with adjustment.

5.3.1 Standard Training Procedure

Let us consider a set of N microscopy scans $\{\mathbf{X}_i\}_{1 \leq i \leq N}$ and corresponding centerline annotations $\{\hat{\mathbf{y}}_i\}_{1 \leq i \leq N}$, in the form of distance maps of the same size as the scans. Voxel p of annotation $\hat{\mathbf{y}}$, denoted $\hat{\mathbf{y}}[p]$, contains the distance from the center of p to the closest centerline. Let $F(\cdot; \Theta)$ be a deep network, with weights Θ . It takes a scan \mathbf{X}_i as input and return a volume $\mathbf{y}_i = F(\mathbf{X}_i; \Theta)$, containing a delineation of centerlines visible in \mathbf{X}_i . To keep the notation concise, we omit the dependencies on \mathbf{y}_i on Θ . The traditional approach to learning the network weights is to make \mathbf{y}_i as close as possible to $\hat{\mathbf{y}}_i$ by solving

$$\Theta^* = \operatorname{argmin}_{\Theta} \sum_{i=1}^N \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i), \quad (5.1)$$

where the loss term $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ measures the voxel-wise difference between the annotation and the prediction. In our experiments, we take \mathcal{L} to be the Mean Square Error. This assumes that the deviations of the annotations from actual centerline trajectories are small and unbiased. In reality, they rarely are. Hence, the network learns to accommodate this uncertainty in the annotations by blurring the predictions. At test time, this leads to breaking the continuity of predictions wherever the image quality is compromised by high level of noise or low contrast between the foreground and the background, as illustrated by Fig. 5.2.

5.3.2 Overview of our Approach

The formulation of Eq. 5.1 assumes that the deviations of the annotations from reality are small and unbiased. This work is predicated on the fact that they rarely are and that we must allow for substantial non-Gaussian deviations from the original annotations. Thus, instead of encoding the annotations in terms of volumes $\hat{\mathbf{y}}_i$, we represent the annotated centerline \mathbf{C}_i of each \mathbf{X}_i as a graph, with the set of vertices \mathcal{V}_i and the set of edges \mathcal{E}_i . Each vertex $v \in \mathcal{V}_i$ has a 3D coordinate c_v , and each edge $(u, v) \in \mathcal{E}_i$ represents a short line segment. This is shown in Fig. 5.2 where the circles along the annotations denote the vertices. Let \mathbf{c}_i be the vector formed by concatenating coordinates of all the vertices of \mathcal{V}_i . To accommodate the possible lack of precision of the annotations, we let \mathbf{c}_i change its initial value. Doing so changes the shape of \mathbf{C}_i but preserves its topology and can be used to explicitly model the deviation of the annotated centerlines from their true position. In other words, the minimization problem can be reformulated as finding

$$\Theta^*, \mathbf{C}^* = \operatorname{argmin}_{\Theta, \mathbf{C}} \sum_{i=1}^N L(\mathbf{c}_i, \mathbf{y}_i) + R(\mathbf{c}_i), \quad (5.2)$$

where $L = \mathcal{L}(D(\mathbf{c}_i), \mathbf{y}_i)$;

\mathbf{C} is the vector obtained by concatenating all the \mathbf{c}_i ; R is a regularization term that forces the deformed centerlines to be smooth, and that we define in Sec. 5.3.3; \mathcal{L} is the same MSE as in Eq. 5.1; and D is a distance transform that creates a volume in which a voxel with coordinates q is assigned its truncated distance to the closest edge of \mathbf{C} . Formally, we write

$$D(\mathbf{c})[q] = \min\{\delta(\mathbf{c}, q), d\}, \quad (5.3)$$

$$\text{where } \delta(\mathbf{c}, q) = \min_{(u,v) \in \mathcal{E}} \min_{0 \leq \phi \leq 1} \|\phi c_u + (1 - \phi)c_v - q\|_2, \quad (5.4)$$

d is the threshold used to truncate the distance map, and the minimization over ϕ serves to find the point on edge (u, v) , that is closest to q .

Solving the problem of Eq. 5.2 means training the network to find centerlines that are smooth and with the same topology as the annotations. This is what we want but, unfortunately, this optimization problem involves two kinds of variables, the components of \mathbf{C} and Θ respectively, which are not commensurate in any way. In practice, this makes optimization difficult. We address this problem by eliminating the \mathbf{C} variables by rewriting Eq. 5.2 as

$$\mathbf{c}_i^*(\mathbf{y}_i) = \operatorname{argmin}_{\mathbf{c}} L(\mathbf{c}_i, \mathbf{y}_i) + R(\mathbf{c}_i), \quad (5.5)$$

$$\Theta^* = \operatorname{argmin}_{\Theta} \sum_{i=1}^N L(\mathbf{c}_i^*(\mathbf{y}_i), \mathbf{y}_i) + R(\mathbf{c}_i^*(\mathbf{y}_i)), \quad (5.6)$$

In the following section, we describe our choice of R and the formulation of $\mathbf{c}_i^*(\mathbf{y}_i)$ that results from it. Eq. 5.6 is a standard continuous optimization problem that we can solve using the usual tools of the trade.

5.3.3 Annotations as Network Snakes

We propose to represent each \mathbf{C}_i as a network snake, and to take R to be a classical sum of spring and elasticity terms [37, 15]. This regularization term takes the form

$$R(\mathbf{c}) = \alpha \sum_{(u,v) \in \mathcal{E}} \|c_u - c_v\|^2 + \beta \sum_{(u,v,w) \in \mathcal{T}} \|c_u - 2c_v + c_w\|^2, \quad (5.7)$$

where α and β are hyper-parameters that balance the strength of the two terms, \mathcal{E} is the set of edges of \mathbf{C} and \mathcal{T} is the set of node triples (u, v, w) such that $(u, v) \in \mathcal{E}$, $(v, w) \in \mathcal{E}$, and v is a node of order two, that is, not a junction of multiple snake branches. As shown in [37, 15], R can be written as

$$R(\mathbf{c}_i) = \frac{1}{2} \mathbf{c}_i^T \mathbf{A} \mathbf{c}_i, \quad (5.8)$$

where A is a sparse symmetric matrix. Given this quadratic formulation of R , we can use the well-known semi-implicit scheme introduced to deform snakes, also known as active contour models [53], to minimize Eq. 5.5. It involves initializing each snake \mathbf{c}_i^0 to the manually produced annotation and refining it by iteratively solving

$$(\mathbf{A} + \gamma \mathbf{I}) \mathbf{c}_i^{t+1} = \gamma \mathbf{c}_i^t - \frac{\partial L}{\partial \mathbf{c}}(\mathbf{c}_i^t, \mathbf{y}_i) \quad (5.9)$$

for \mathbf{c}_i^{t+1} , where γ is a hyper-parameter known as the *viscosity* and is inversely proportional to the step size in each iteration. We refer the reader to [53] for the complete derivation. Here we only note, that when the iteration stabilizes, we have $\forall i, \mathbf{c}_i^t \approx \mathbf{c}_i^{t+1}$. We can therefore denote the stable vector of node locations by \mathbf{c}_i^* , substitute $\mathbf{c}_i^{t+1} \approx \mathbf{c}_i^t \approx \mathbf{c}_i^*$ in Eq. 5.9, and use the derivative of Eq. 5.8, to write

$$\forall i, \quad \frac{\partial R}{\partial \mathbf{c}}(\mathbf{c}_i^*) + \frac{\partial L}{\partial \mathbf{c}}(\mathbf{c}_i^*, \mathbf{y}_i) \approx 0, \quad (5.10)$$

which means that \mathbf{c}^* minimizes $R + L$ and is a solution of Eq. 5.5.

In practice, we solve Eq. 5.9 by inverting the matrix $(\mathbf{A} + \gamma \mathbf{I})$ at the start of the training procedure and then multiplying the right-hand-side of the equation by the inverse at each iteration. Hence, we write

$$\mathbf{c}_i^{t+1} = (\mathbf{A} + \gamma \mathbf{I})^{-1} (\gamma \mathbf{c}_i^t - \frac{\partial L}{\partial \mathbf{c}}(\mathbf{c}_i^t, \mathbf{y}_i)). \quad (5.11)$$

We perform the update of Eq. 5.11 for $0 \leq t < T$. We take $T = 10$ in our implementation, which is sufficient for the process to stabilize, and denote the result of the last iteration by $\mathbf{c}_i^*(\mathbf{y}_i) = \mathbf{c}_i^T$.

5.3.4 Computing the Gradients of the Loss Function

Performing the minimization in Eq. 5.6 requires computing at each iteration the gradient of the loss with respect to the network output \mathbf{y}_i . To avoid cluttering the notation, we denote

$\mathbf{c}^*(\mathbf{y}_i)$ by \mathbf{c}^* . The gradient can then be expressed as

$$\begin{aligned} & \frac{\partial}{\partial \mathbf{y}} (L(\mathbf{c}_i^*, \mathbf{y}_i) + R(\mathbf{c}_i^*)) \\ &= \frac{\partial L}{\partial \mathbf{y}}(\mathbf{c}_i^*, \mathbf{y}_i) + \left(\frac{\partial L}{\partial \mathbf{c}}(\mathbf{c}_i^*, \mathbf{y}_i) + \frac{\partial R}{\partial \mathbf{c}}(\mathbf{c}_i^*) \right) \frac{\partial \mathbf{c}_i^*}{\partial \mathbf{y}} \\ &\approx \frac{\partial L}{\partial \mathbf{y}}(\mathbf{c}_i^*, \mathbf{y}_i), \end{aligned} \quad (5.12)$$

where we used Eq. 5.10 to eliminate the second term. In other words, even though \mathbf{c}^* is a function of \mathbf{y}_i , we do not need to compute its derivatives with respect to \mathbf{y}_i to train the neural network. We only need those of L , and can treat \mathbf{c}^* as a constant when evaluating them. Therefore, the only difference between using our approach and the standard one of Section 5.3.1 is that instead of evaluating the loss using the original annotation \mathbf{c} , we use its optimized version \mathbf{c}^* . We call this approach *SnakeFull* and it is depicted at the top of Fig. 5.3.

5.3.5 Speeding Things Up

We will show in Section 5.4 that *SnakeFull* performs well but is slow to train. The culprit is the term $\frac{\partial L}{\partial \mathbf{c}}$ in the update Eq. 5.11, which involves a time-consuming computation of the gradient of a distance map. To speed things up, we introduce a faster approach that we call *SnakeFast*. In it, we replace the term L in Eq. 5.5 by a simpler objective function S directly inspired by the classical external snake energy [53]. We take it to be

$$S(\mathbf{c}, \mathbf{y}) = \sum_{v \in \mathcal{V}} (\mathbf{y} * G)[c_v], \quad (5.13)$$

where $*G$ denotes a convolution with a Gaussian kernel and $\mathbf{y}[c_v]$ denotes the network output at vertex v . S is very similar to the energies used in traditional network snake formulations [37, 15]. Importantly, S and its gradients are easy and fast to compute because doing so only requires convolving \mathbf{y} with a Gaussian kernel and sampling the result at the locations of the snake nodes. Deforming the annotations then involves finding

$$\mathbf{c}_i^\dagger(\mathbf{y}_i) = \operatorname{argmin}_{\mathbf{c}} S(\mathbf{c}, \mathbf{y}_i) + R(\mathbf{c}), \quad (5.14)$$

which means that the sum of distance values along the snake should be as low as possible while preserving snake smoothness. As in Section 5.3.3, the snake update takes the form

$$\mathbf{c}_i^{t+1} = (\mathbf{A} + \gamma \mathbf{I})^{-1} (\gamma \mathbf{c}_i^t - \frac{\partial S}{\partial \mathbf{c}}(\mathbf{c}_i^t, \mathbf{y}_i)). \quad (5.15)$$

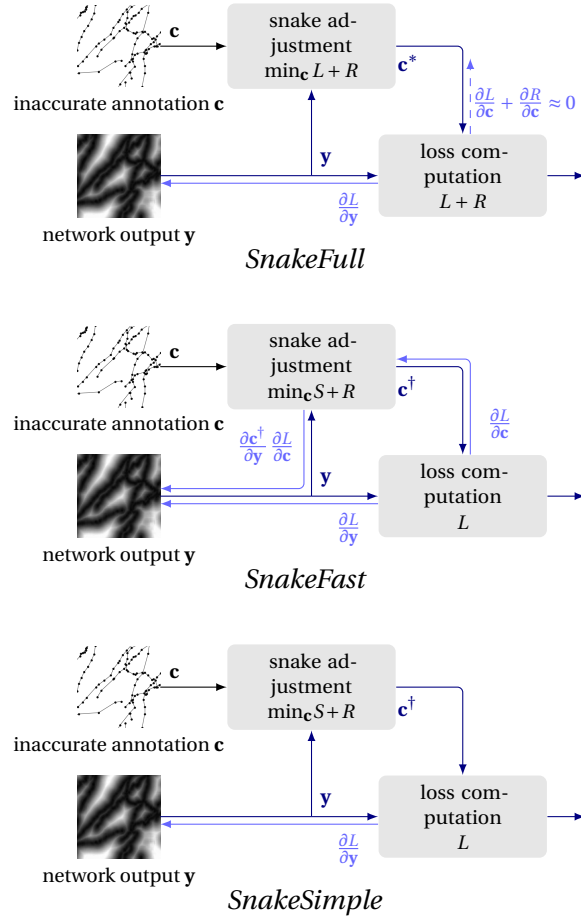


Figure 5.3: The three approaches to training described in Sec. 5.3.4 and 5.3.5. In *SnakeFull*, the training objective is also used as the objective of the snake. This makes some gradient components vanish, simplifying gradient computation, but results in snake updates that are costly to compute. *SnakeFast* can accommodate an arbitrary snake objective, which makes it faster than *SnakeFull*, even though it requires backpropagation through a sequence of snake updates. In *SnakeSimple*, the backpropagation over the snake updates is simply omitted. This approach is the fastest. We analyze the accuracy vs. speed tradeoff induced by these three methods in section 5.4.

In practice, we take $\mathbf{c}_i^\dagger(\mathbf{y}_i) = \mathbf{c}_i^T$, where $T = 10$, as in Section 5.3.3. Finally, we take the network training objective to be

$$\Theta^* = \operatorname{argmin}_{\Theta} \sum_i L(\mathbf{c}_i^\dagger(\mathbf{y}_i), \mathbf{y}_i), \quad (5.16)$$

where we still use the original L of Eq. 5.2. We do this because S only depends on a small subset of voxels of \mathbf{y} . Hence, it only provides a sparse supervisory signal and is not well suited as the training objective for the network that produces a dense distance map. The gradient of

Chapter 5. Adjusting The Ground Truth Annotations for Connectivity-Based Learning to Delineate

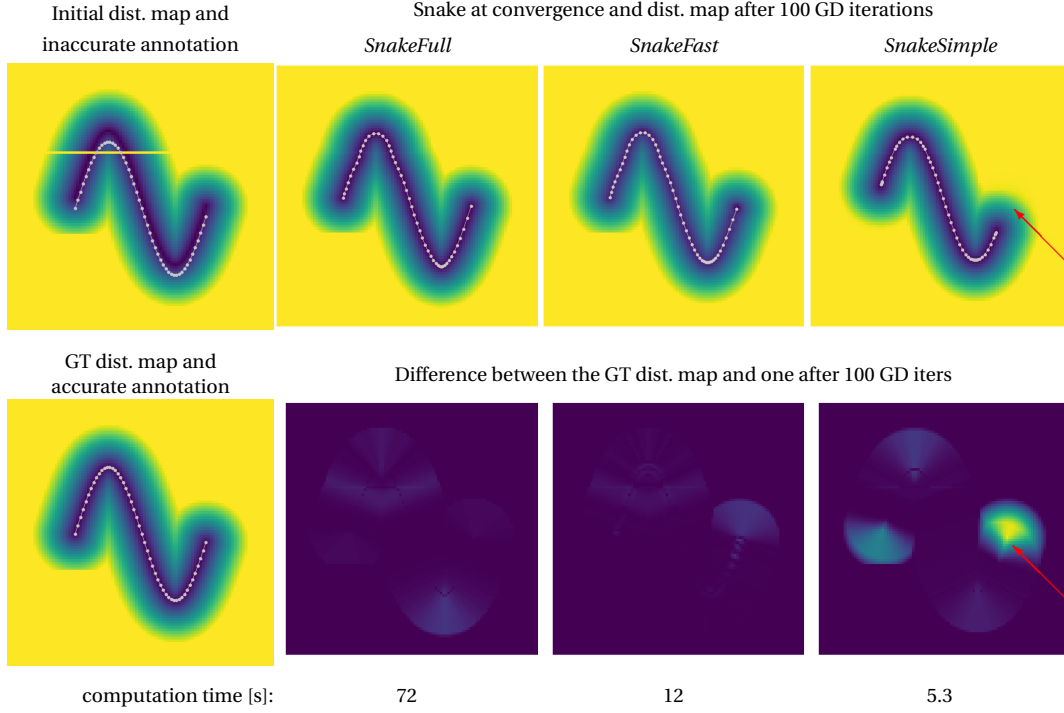


Figure 5.4: **Compared behavior of *SnakeSimple*, *SnakeFast*, and *SnakeFull* on a synthetic 2D example.** (Left column) At the bottom, distance map and corresponding annotation. At the top, we simulated an unwarranted break in the distance map (horizontal yellow line) and shifted the annotation by several pixels. (Other Columns) In three separate runs, we performed 100 Gradient Descent using either *SnakeFull*, *SnakeFast*, or *SnakeSimple*. In the top row, we show the corrected annotation and the updated distance maps. The bottom row depicts the differences between the updated maps and the ground-truth one. We also indicate the computation times. *SnakeFull* removes the interruption in the distance map but the computation is slow. *SnakeFast* is much faster and fills the gap in the distance map almost as well. *SnakeSimple* is even faster but yields a corrected annotation that is too short, as highlighted by the red arrow.

the objective of Eq. 5.16 is

$$\frac{\partial}{\partial \mathbf{y}} L(\mathbf{c}^\dagger, \mathbf{y}) = \frac{\partial L}{\partial \mathbf{y}}(\mathbf{c}^\dagger, \mathbf{y}) + \frac{\partial L}{\partial \mathbf{c}}(\mathbf{c}^\dagger, \mathbf{y}) \frac{\partial \mathbf{c}^\dagger}{\partial \mathbf{y}}. \quad (5.17)$$

Because we minimized S instead of L in Eq. 5.14, we can no longer assume that the second term is zero as we did in Section 5.3.4. Hence, to compute it during the minimization, we backpropagate through the snake update procedure of Eq. 5.15, as depicted by the middle row of Fig. 5.3. In practice, we use the autograd functionality of Pytorch to this end.

The non-zero second term of Eq. 5.17 helps guide the snake to a position where the data loss L is low and ultimately influences the distance map that our deep network F outputs. It could be argued that ignoring this term so that the networks focuses exclusively on fitting the annotations would be preferable. To test this assertion, we implemented *SnakeSimple*, a third variant of our approach in which we take the second term of Eq. 5.17 to be zero. *SnakeSimple*

is even faster than *SnakeFast*. In essence, it is a simplified version of *SnakeFull* and *SnakeFast* in which we successively optimize the network weights and then the snake position without any direct interaction between these two optimization steps.

Fig. 5.4 uses a synthetic example to illustrate the differences between our three variants. *SnakeFast* and *SnakeFull* yield similar results with the former being much faster whereas *SnakeSimple* is even faster but prone to generating artifacts. We now turn to our experimental results on real data that confirm this.

5.4 Experiments

5.4.1 Datasets

We tested our approach on the following data sets.

- The *Neurons* data set comprises fourteen two-photon microscopy 3D scans of fragments of a mouse brain, with manually traced neurites. We use four volumes for testing and ten for training, each of size $200 \times 250 \times 250$ voxels and spatial resolution $0.3 \times 0.3 \times 1.0 \mu\text{m}$.
- The *Brain* data set contains two 3D images of neurons in a mouse brain. They had been outlined manually while viewing the sample under a microscope and the image was captured later. The sample deformed in the meantime, exacerbating misalignment between the annotation and the image. We use one stack of size $151 \times 714 \times 865$ voxels and a resolution of $1 \mu\text{m}$ for training and one of size $228 \times 764 \times 1360$ for testing.
- The *MRA* is a publicly available set of Magnetic Resonance Angiography brain scans [14]. It consists of 42 annotated stacks, which we cropped to $416 \times 320 \times 128$ voxels by removing their empty margins. Their resolution is $0.5 \times 0.5 \times 0.6 \text{ mm}$. We randomly partitioned the data into 31 training and 11 test volumes.

None of our data sets can be considered as perfectly annotated. All annotations were performed as accurately as possible, but their precision is affected by the uneven distribution of the dye, image noise, and generic difficulty of annotating 3D volumes. In *Brain*, the difficulty is compounded by the fact that the annotation were performed live days before image acquisition, and the sample deformed in the meantime.

5.4.2 Metrics

We used the following performance metrics.

- CCQ. Since standard segmentation metrics such as the F1 score [82] and precision-recall break-even point [68] are very sensitive to misalignment of thin structures, we

use the *correctness-completeness-quality*, which is specifically designed for linear structures [100]. Correctness corresponds to precision, completeness to recall, and quality to the intersection-over-union. However, the notion of a true positive is relaxed from perfect coincidence of the ground truth and the prediction to their co-occurrence within a distance of d pixels. We used $d = 3$. Although it accounts for possible ground truth misalignment, *CCQ* is still a voxel-wise metric, insensitive to topological errors, such as short interruptions of neurites.

- *APLS*. The *Average Path Length Similarity* is defined as the aggregation of relative length differences of shortest paths between pairs of corresponding end points, randomly sampled in the reconstructed and predicted graphs. It was introduced to evaluate road map reconstructions from aerial images [92] and aims to evaluate the connectivity of the reconstructions, as opposed to their pixel-wise accuracy, which makes it a perfect performance measure for our task.
- *TLTS*. The *Too-Long-Too-Short* is another performance criterion based on statistics of relative lengths of shortest paths between corresponding pairs of end points in the prediction and the ground truth [99]. We report the fraction of *correct* paths, that is, predicted paths whose relative length difference to the corresponding ground truth paths is lower than 15%.

5.4.3 Architectures and Training Details

Our contribution lies in the updating of the annotations and the loss function we use to achieve it, which should improve performance independently of any specific network architecture. To demonstrate this, we used two different architectures.

- *UNet*. A 3D *UNet* [81] with three max-pooling layers and two convolutional blocks. The first layer has 64 filters. Each convolution layer is followed by a batch-normalization and dropout with a probability of 0.15. During training, we randomly crop sub-volumes of size $96 \times 96 \times 96$ and flip them along each dimension with probability 0.5. We combine them into batches of 8.
- *DRU*. A recurrent architecture iteratively refining segmentation output 3 times [98]. The first layer has 64 filters. Each convolution layer is followed by a group-normalization and dropout with a probability of 0.15. During training, we randomly crop sub-volumes of size $96 \times 96 \times 96$ and flip them along each dimension with probability 0.5. We combine them into batches of 4. To compute the loss function, we average the outputs of all 3 refinement steps. During testing, the output of the final step is used to evaluate performance.

We trained both architectures in four different ways: by minimizing the Mean Squared Error to the original annotations, which we will refer to as *OrigAnnot*, and by using the *SnakeSimple*,

Table 5.1: Performance of deep nets trained with different loss functions on our three data sets and the time needed for single training iteration.

	Method	Pixel-wise			Topology-aware		iter. t.
		<i>Corr.</i>	<i>Compl.</i>	<i>Qual.</i>	<i>APLS</i>	<i>TLTS</i>	s
<i>Neurons</i>	<i>UNet-OrigAnnot</i>	98.9	91.3	90.4	80.3	80.9	2.8
	<i>UNet-SnakeSimple</i>	98.4	92.5	91.2	84.2	83.4	3.8
	<i>UNet-SnakeFull</i>	99.0	94.4	93.5	89.3	85.9	18.9
	<i>UNet-SnakeFast</i>	98.7	95.0	93.8	91.1	85.9	5.2
	<i>DRU-OrigAnnot</i>	97.2	94.0	91.5	84.3	83.9	2.7
	<i>DRU-SnakeSimple</i>	97.4	95.2	92.9	90.8	85.9	3.8
	<i>DRU-SnakeFull</i>	96.9	96.9	94.1	91.8	89.3	19.1
	<i>DRU-SnakeFast</i>	97.0	97.1	94.2	91.7	88.1	5.3
	<i>NR-Dice</i>	97.7	97.0	94.8	81.0	83.6	2.8
	<i>QAM</i>	94.5	98.8	93.5	87.3	84.5	4.2
	<i>DS6</i>	97.5	97.0	94.7	83.8	84.1	5.8
<i>Brain</i>	<i>UNet-OrigAnnot</i>	81.8	83.5	70.4	65.8	63.6	2.8
	<i>UNet-SnakeSimple</i>	83.0	83.9	71.6	70.4	68.8	3.4
	<i>UNet-SnakeFull</i>	83.5	85.4	73.1	74.2	69.9	17.8
	<i>UNet-SnakeFast</i>	83.1	85.5	72.9	73.9	70.2	4.9
	<i>DRU-OrigAnnot</i>	82.1	86.5	72.8	68.9	69.5	2.7
	<i>DRU-SnakeSimple</i>	83.2	87.7	74.5	73.8	74.6	3.5
	<i>DRU-SnakeFull</i>	84.4	88.5	76.1	74.8	78.1	18.3
	<i>DRU-SnakeFast</i>	84.2	88.9	76.2	75.1	77.7	5.1
	<i>NR-Dice</i>	85.2	83.4	72.8	67.6	65.2	2.8
	<i>QAM</i>	89.8	79.3	72.8	71.2	68.6	4.2
	<i>DS6</i>	83.2	81.6	70.0	71.0	68.8	5.8
<i>MRA</i>	<i>UNet-OrigAnnot</i>	90.1	72.2	66.9	49.8	50.4	2.8
	<i>UNet-SnakeSimple</i>	89.9	73.1	67.5	53.5	53.1	3.7
	<i>UNet-SnakeFull</i>	90.2	73.5	68.0	55.6	55.0	18.5
	<i>UNet-SnakeFast</i>	90.3	73.5	68.1	55.4	55.2	5.1
	<i>DRU-OrigAnnot</i>	80.2	79.3	66.3	48.7	49.9	2.7
	<i>DRU-SnakeSimple</i>	80.7	79.9	67.1	53.3	53.0	3.7
	<i>DRU-SnakeFull</i>	80.9	80.5	67.6	55.6	55.2	18.8
	<i>DRU-SnakeFast</i>	81.0	80.5	67.7	55.3	55.4	5.2
	<i>NR-Dice</i>	85.5	77.3	68.3	50.2	53.8	2.8
	<i>QAM</i>	80.2	80.1	66.8	54.3	54.2	4.2
	<i>DS6</i>	82.0	80.3	68.1	55.0	54.9	5.8

SnakeFull, and *SnakeFast* variants of our approach, described in Sections 5.3.4 and depicted by Fig. 5.3. In all cases, we used Adam [54] with the learning rate set to $1e-4$, and a weight decay

Chapter 5. Adjusting The Ground Truth Annotations for Connectivity-Based Learning to Delineate

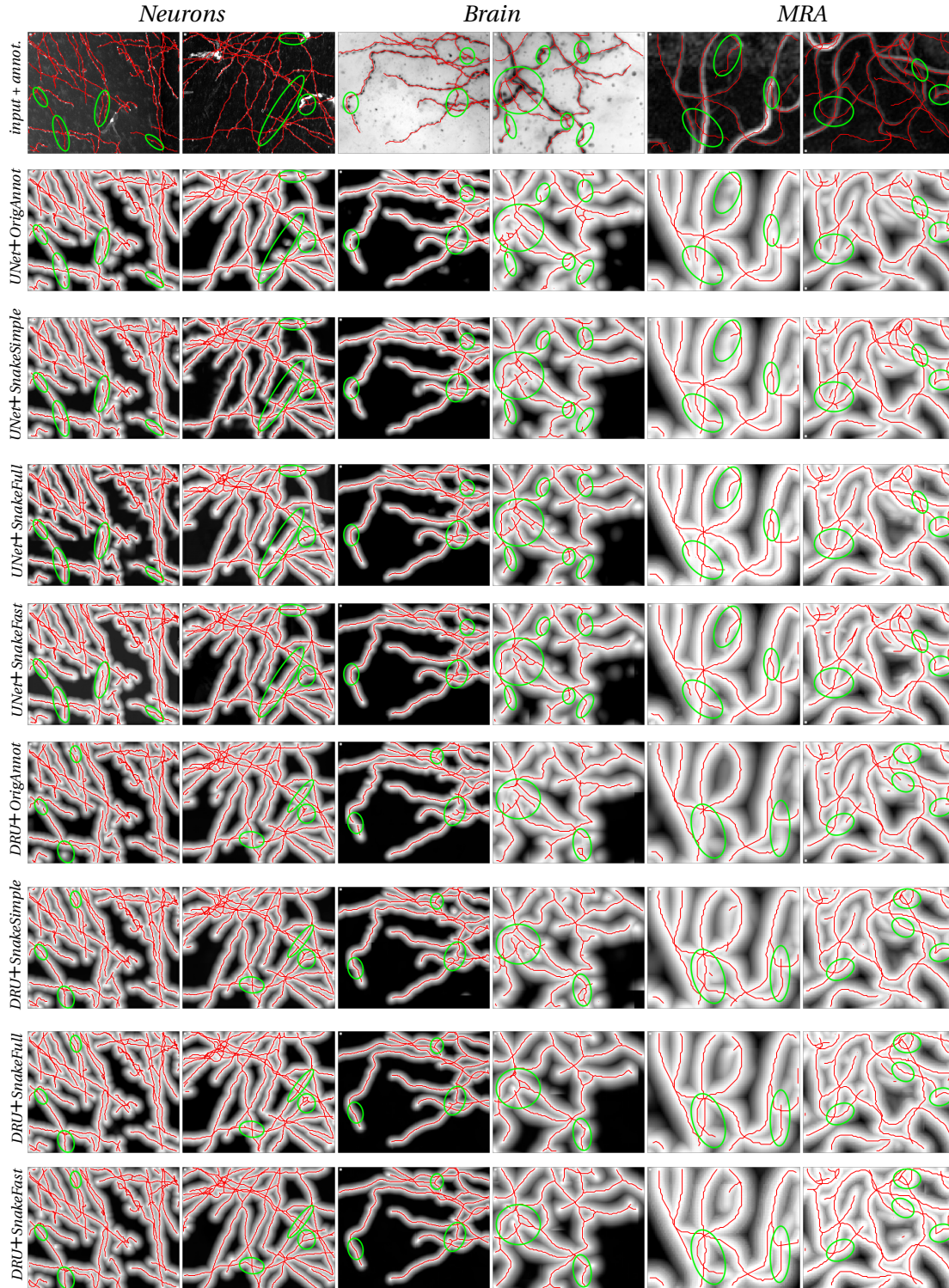


Figure 5.5: Test predictions of different methods on three data sets. The green ellipses denote areas where training with the original annotations results in unwarranted breaks in the delineations whereas our approach does not.

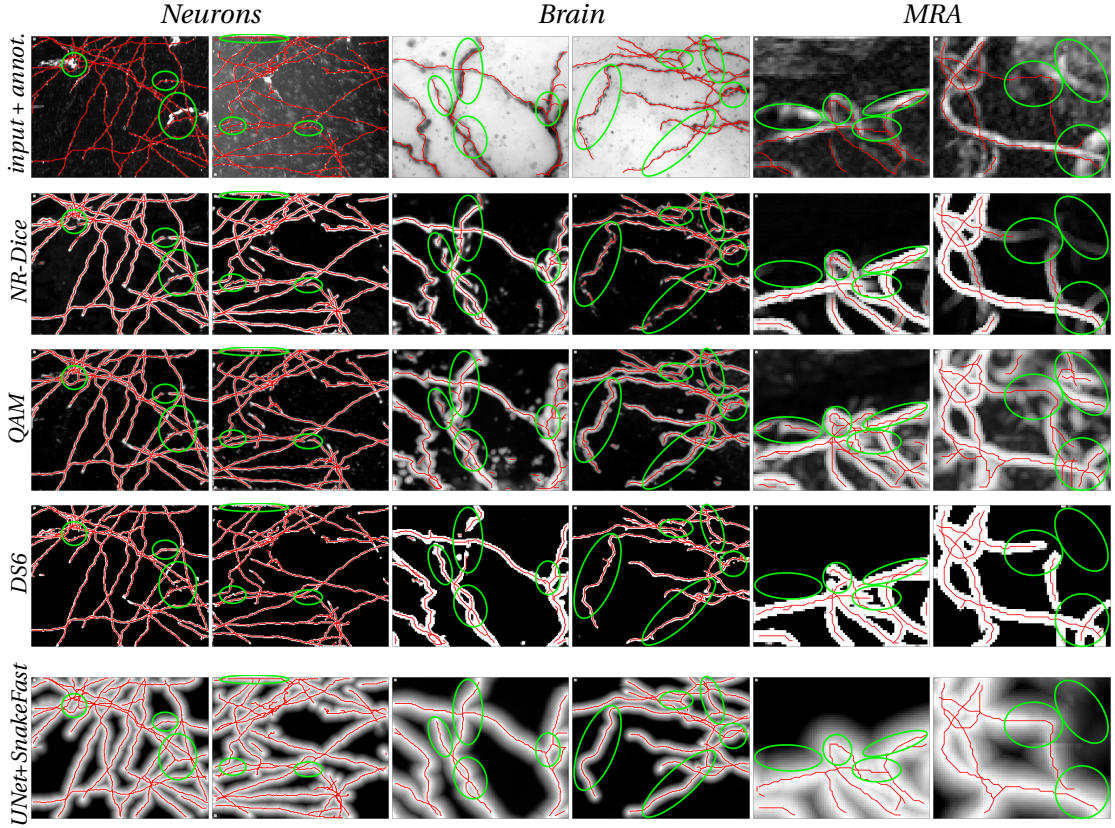


Figure 5.6: Qualitative comparison of the results of *SnakeFast* to existing methods of training with noisy labels. The green ellipses denote areas where baselines result in unwarranted breaks in the delineations at test time whereas our approach does not.

of $1e-4$. At test time, the predicted distance map were thresholded at 2 and skeletonized to obtain centerlines. To compute the *TLTS* and *APLS* scores, we converted them into graphs.

5.4.4 Label Correction Baselines

As noted in section 5.2, we do not know of other methods that deform the annotation graph during training, while maintaining its topology. However, there are methods designed to train deep nets using noisy annotations, where the noise is understood as flipping some pixel labels. In the following section, we compare our algorithm to three such methods:

- *NR-Dice*. A *UNet* trained with the Noise Robust Dice Loss proposed in [97].
- *QAM*. An architecture with an auxiliary deep network to recognize annotations that might be wrong and downplay their importance during training [107].
- *DS6*. A Siamese architecture and a training routine dedicated to enforcing equivariance of the network to deformations [20].

Chapter 5. Adjusting The Ground Truth Annotations for Connectivity-Based Learning to Delineate

Table 5.2: Performance of a *UNet* trained with the *OrigAnnot* and with *SnakeFast* on the *Synthetic* data set with precise annotations.

Arch.	Method	Pixel-wise			Topology-aware		iter. t.
		<i>Corr.</i>	<i>Compl.</i>	<i>Qual.</i>	<i>APLS</i>	<i>TLTS</i>	s
<i>UNet</i>	<i>OrigAnnot</i>	86.9	86.5	77.2	92.8	89.0	2.8
	<i>SnakeFast</i>	86.6	86.7	77.1	93.2	89.3	4.8

5.4.5 Comparative Evaluation.

We present example reconstructions in Fig. 5.5 and Fig. 5.6. As shown in Tab. 5.1, *SnakeFull* and *SnakeFast* outperform *OrigAnnot* in *CCQ* terms by a small margin, and in *APLS* and *TLTS* terms by a significantly larger one, which confirms that the main benefit of our loss is the improved connectivity of the predictions. As can be seen in Fig 5.5, our approach to training yields delineations with fewer unwarranted breaks and longer uninterrupted curvilinear segments.

On average *UNet* and *DRU* perform best when trained with *SnakeFull* and *SnakeFast*. However, *SnakeFast* requires three times less time per training iteration. *SnakeSimple* delivers a further 20-30% speedup but incurs a clear performance drop. Crucially, these conclusions apply to both the *UNet* and *DRU* architectures. In fact, the performance gain resulting from switching from *OrigAnnot* to *SnakeFast* is larger than the one resulting from changing from the simpler *UNet* to the more sophisticated *DRU* while retaining the standard *OrigAnnot* approach to training.

In short, *SnakeFast* represents an excellent compromise between training time and performance. This being said, at test time, the run-time is the same no matter how the network was trained, because there is no alignment of annotations anymore. Hence, given sufficient computational resources, *SnakeFull* is also a valid option.

The bottom third of each part of Tab. 5.1 measures the performance of the methods designed to accommodate label noise, as described in Section 5.4.4. Because they don't explicitly preserve annotation topology and we do, *UNet* trained with *SnakeFast* outperform these methods in terms of the topology-aware scores but not necessarily in terms of the pixel-aware ones, which are note our main concern.

5.4.6 Perfectly Accurate Annotations

Having demonstrated that our loss function improves delineation results when the annotations lack spatial precision in Sec. 5.4.5, we now investigate its behavior when the annotation is precise. Since it is virtually impossible to precisely annotate 3D microscopy scans, we resort to synthetic data set *Synthetic*, which we generated using the *VascuSynth* algorithm [46, 52] and its implementation [105]. The images are generated from vascular graphs, which we use as

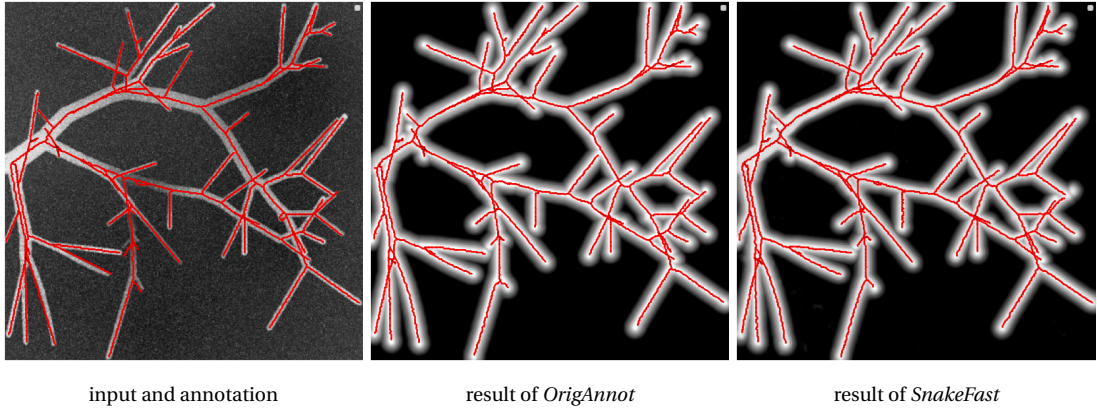


Figure 5.7: Results of training with the precise annotations of the *Synthetic* data set. When the annotations are precise, *SnakeFast* performs as well as training with the *OrigAnnot*.

perfectly accurate annotations. We used twenty stacks for training and ten for testing, each of size $400 \times 400 \times 400$. Fig. 5.7 shows the maximum-intensity projection of a test stack. The results, presented in Tab. 5.2, confirm that, for perfectly accurate annotations, our method reduces to standard training with the MSE without incurring any performance drop.

5.4.7 Increasing Annotation Inaccuracy

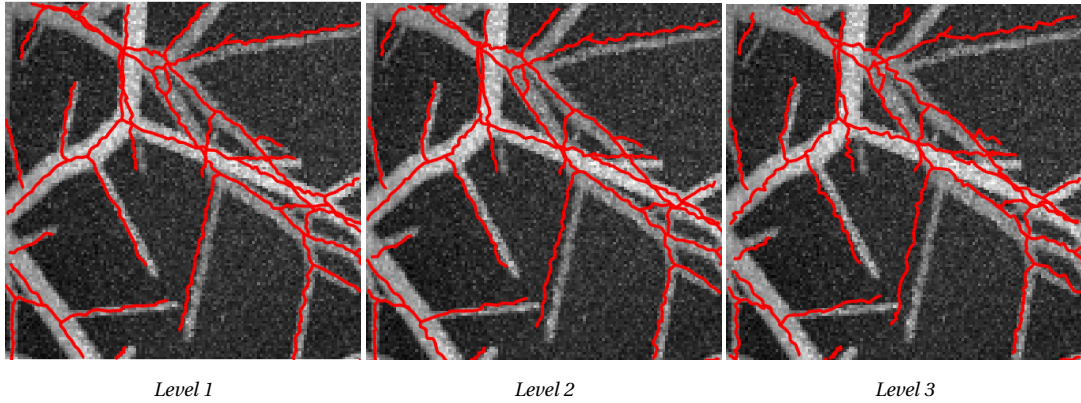


Figure 5.8: **Annotation Deformation Levels.** The deformation magnitude increases from left to right.

To investigate how increasing the level of inaccuracy of the annotations affects the performance of a *UNet* trained with *SnakeFast*, we perturbed the annotations of the *Synthetic* data set. We applied a random deformation field that varies slowly across space to each annotation graph. We modulated its amplitude to change the level of inaccuracy. This produced three sets of annotations, as depicted by Fig. 5.8. We trained the network on each of them and present the results in Fig. 5.9. When the network is trained with *SnakeFast*, its connectivity-related scores degrade much slower than when trained using *OrigAnnot*.

Chapter 5. Adjusting The Ground Truth Annotations for Connectivity-Based Learning to Delineate

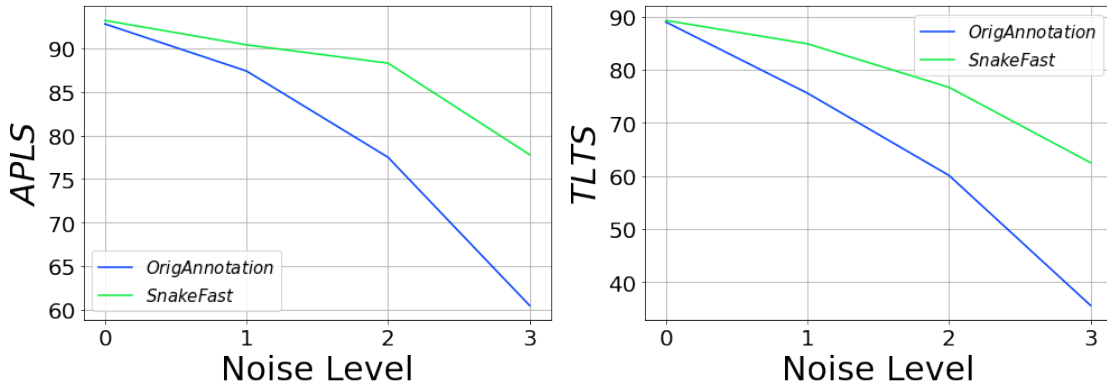


Figure 5.9: **Increasing the amount of deformation.** *APLS* and *TLTS* scores as a function of the deformation level. The *OrigAnnot* scores decrease fast whereas those of *SnakeFast* decrease much more slowly.

Table 5.3: Performance of *UNet* trained using *SnakeFast* and *OrigAnnot* on the *Neurons* data set with very coarse annotations. Performance of *UNet* trained using the precise annotations shown for reference.

Annot.	Method	Pixel-wise			Topology-aware	
		<i>Corr.</i>	<i>Compl.</i>	<i>Qual.</i>	<i>APLS</i>	<i>TLTS</i>
coarse	<i>OrigAnnot</i>	85.2	67.6	60.4	46.5	50.9
	<i>SnakeFast</i>	97.6	87.0	85.3	66.8	73.5
precise	<i>OrigAnnot</i>	98.9	91.3	90.4	80.3	80.9
	<i>SnakeFast</i>	98.7	95.0	93.8	91.1	85.9

5.4.8 Reducing Annotation Effort

The robustness of *SnakeFast* to deviations in the annotation inspired us to ask another question: Can this loss function be used to train deep networks with annotations that are simplified to the point where they become much easier, faster, and therefore cheaper to obtain? To answer this, we trained the *UNet* with *SnakeFast* and *OrigAnnot* on the *Neurons* data set with very coarse annotations. We obtained them by connecting neurite branching- and end-points with straight lines, as shown in Fig. 5.10. The results are presented in Tab. 5.3. As expected, training on the coarse annotations without adjusting them results in a significant performance drop as compared to training on precise annotations. Switching from precise to coarse annotations still incurs a performance drop when using *SnakeFast*, but a much smaller one than when using the baseline. Visual inspection of the resulting segmentations, shown in Fig. 5.11, leads us to conclude that, for tasks where a compromise between accuracy and annotation cost is acceptable, using the easy annotations together with *SnakeFast* is a viable alternative to the classical approach.

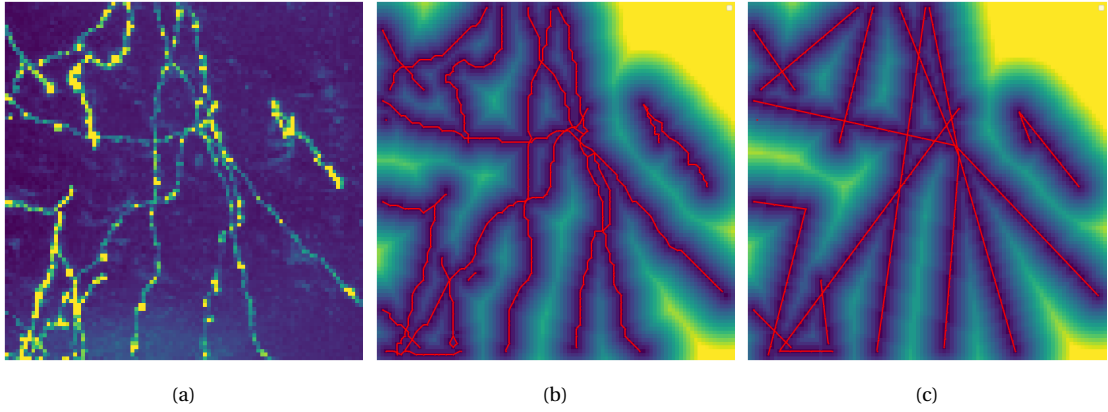


Figure 5.10: **Coarse Annotations** (a) Training image of a neurite (b) Distance map obtained from original annotation overlaid in red (c) Distance map obtained from coarse annotation overlaid in red. Coarse annotations are obtained by connecting neurite end points and bifurcations with straight lines, and are easier to perform than full annotations.

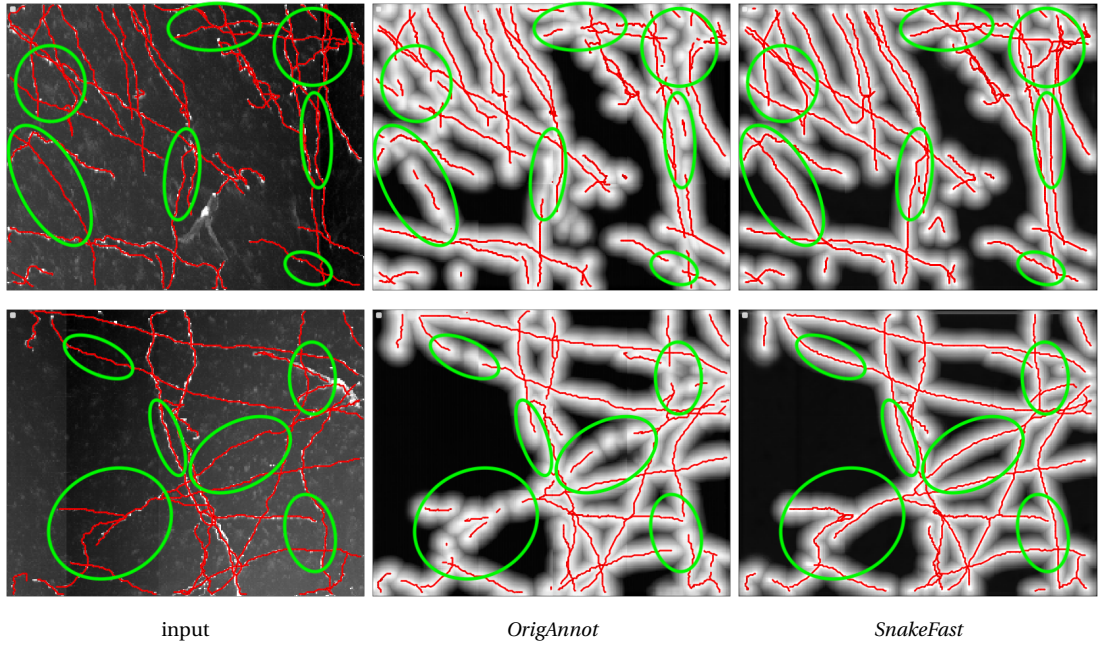


Figure 5.11: Results of training a *UNet* with *OrigAnnot* and *SnakeFast* on the *Neurons* data set with easy annotations.

5.4.9 Ablation Study

To investigate the impact of hyper-parameters of our method on performance, we run the following ablation studies.

Chapter 5. Adjusting The Ground Truth Annotations for Connectivity-Based Learning to Delineate

Table 5.4: Performance of *UNet* trained using *SnakeFast* on the *Neurons* data set when varying the elasticity and spring term coefficients.

		Pixel-wise			Topology-aware		iter. t.
		<i>Corr.</i>	<i>Compl.</i>	<i>Qual.</i>	<i>APLS</i>	<i>TLTS</i>	s
$\alpha = 1e-2$	$\beta = 1e-4$	99.0	94.5	93.5	88.1	84.8	5.2
	$\beta = 1e-3$	98.7	95.0	93.8	91.1	85.9	5.2
	$\beta = 1e-2$	98.4	94.0	92.7	85.1	84.3	5.2
	$\beta = 1e-1$	98.9	93.8	92.8	83.8	84.1	5.2
$\alpha = 1e-4$		98.4	92.9	91.5	86.6	83.0	5.2
$\alpha = 1e-3$		99.0	94.2	93.4	85.3	84.4	5.2
$\alpha = 1e-2$		98.7	95.0	93.8	91.1	85.9	5.2
$\alpha = 1e-1$		98.7	94.3	93.1	79.8	82.5	5.2

Table 5.5: Performance of *UNet* trained using *SnakeFast* on the *Neurons* data set when varying the inverse stepsize, together with the number of snake updates used in every training iteration and the resulting iteration time.

	Pixel-wise			Topology-aware		no steps	iter. t.
	<i>Corr.</i>	<i>Compl.</i>	<i>Qual.</i>	<i>APLS</i>	<i>TLTS</i>		s
$\gamma = 100$	98.8	94.5	93.4	90.9	85.8	80	6.3
$\gamma = 10$	98.7	95.0	93.8	91.1	85.9	10	5.2
$\gamma = 1$	— the snake diverged —					10	5.2

Regularization terms

The regularization term R of Eq. 5.7 is the sum of a spring term, weighted by a coefficient α , and an elasticity term, weighted by a coefficient β . To investigate their influence on performance, we varied α and β and trained our *UNet* on the *Neurons* data set. The results are presented in Tab. 5.4. The best results are attained with relatively low values of both terms. Higher values of the spring term, originally proposed for closed contours, effectively regularize loopy topologies, but when used on tree-shaped structures, representing blood vessels and neuronal processes, tend to shorten the reconstructed neurites and vessels. Higher values of the elasticity term make it more difficult to fit irregular trajectories of neurites, like the ones shown in Fig. 5.5.

Step size for snake update

As explained in section 5.3.3, the snake update iteration has a parameter γ , called viscosity, that acts as an inverse step size. We report the results of changing γ in Tab. 5.5. Low viscosity results in large step size and can make the snake update procedure diverge, which we observed

Table 5.6: Performance of deep nets trained with *MAE* and *MSE* costs on the *Neurons* data set and the time needed for single training iteration.

Cost	Method	Pixel-wise			Topology-aware		iter. t. s
		<i>Corr.</i>	<i>Compl.</i>	<i>Qual.</i>	<i>APLS</i>	<i>TLTS</i>	
<i>MAE</i>	<i>OrigAnnot</i>	98.6	91.2	90.1	81.4	80.5	2.8
	<i>SnakeFast</i>	98.8	94.6	93.4	89.9	85.8	5.2
<i>MSE</i>	<i>OrigAnnot</i>	98.9	91.3	90.4	80.3	80.9	2.8
	<i>SnakeFast</i>	98.7	95.0	93.8	91.1	85.9	5.2

for $\gamma = 1$. On the other hand, high viscosity corresponds to small step size and increases the risk that the snake does not converge within the preset number of iterations. With $\gamma = 100$, we needed to increase the number of snake updates from 10 to 80 to ensure convergence. This also increased the iteration time by one second. $\gamma = 10$ made the snake converge within 10 updates, while also resulting in marginally higher performance than $\gamma = 100$.

L1 vs L2 distance

We also verified the performance of a *UNet* trained with *SnakeFast* when changing the loss data term from Mean Squared Error to Mean Absolute Error. The results, shown in Tab. 5.6 show very slight advantage of *MSE*, possibly due to a gradient profile that prioritizes penalizing higher errors.

5.5 Conclusion

We have proposed a method that accounts for the inevitable inaccuracies in manual annotations of curvilinear 3D structures, such as neurites and blood vessels, in 3D image stacks. It leverages on the network snake formalism to define a loss function that simultaneously trains the deep network to produce the delineation and adjusts the initially imprecise annotations.

Our approach does not depend on the specific network architecture we use. Hence, its effectiveness suggests that handling such imprecisions may be even more important than refining the network architecture, which is something that has been largely neglected in the literature.

In future work, we will investigate the extension our approach to segmenting surfaces, like cell membranes in electron microscopy scans.

6 Conclusion

In this thesis, we propose topology-aware methods that reduces topological errors in reconstruction of curvilinear structures from 2D and 3D images. In this chapter, we briefly present a summary of the methods proposed in this thesis and discuss future research directions.

6.1 Summary

In chapter 2, we propose a differentiable loss function that successfully enforces proper connection on the output of binary segmentation ConvNets for the purpose of delineating the boundaries of the road and irrigation canal networks. Using this loss function to train a simple UNet allows us to outperform considerably more sophisticated architectures on hard benchmark datasets. This implies that we might not have fully realized the potential of these simpler networks, and that one way to do so might be to introduce suitable constraints during training.

In chapter 3, we showed a flaw in the way that existing techniques to train deep networks to delineate curvilinear structures using persistent homology are designed: by using insufficient filtration functions, they drastically reduce the information content of the persistence diagrams, which negatively affects the trained network's performance. With our new method, which combines filtration by thresholding and the height function, the persistence diagrams' descriptive power is increased, and Persistent Homology is now one of the most effective techniques for training topologically accurate deep networks.

In chapter 4, we have proposed a method that takes into account the inevitable inaccuracies that are present in the manual annotation of curvilinear structures in 3D image stacks. Some examples of these types of structures include neurites and blood vessels. We tackle this issue by training the deep network to produce the delineation while concurrently adjusting the annotations that were previously too imprecise. This is accomplished by leveraging the network snake formalism, which defines a loss function. We demonstrate that the proposed method outperforms baselines. Furthermore, we show that a deep network can be trained

even with coarse annotations, hence, significantly reducing the annotation cost.

In chapter 5, we extend our work presented in chapter 2, so that it can work on 3D data as well. We propose a loss function that enforces topological consistency using 2D projections. We apply our topology-aware loss on multiple projections of predictions. When a deep network is trained with our loss, the 3D connectivity of its outputs is significantly improved, and the amount of annotation work that is required to obtain training data is significantly reduced.

Our work was aimed at ensuring correct connectivity of delineations produced by deep learning algorithms. Connectivity is an inherently non-local property. As such, it cannot be enforced by loss function like the cross entropy and the mean squared error, that evaluate marginal distributions of the predicted variables. We therefore focused on developing new, non-local, connectivity-oriented loss functions.

The main challenge associated to this task stems from the difficulty of evaluating the connectivity in a way that is at the same time accurate, efficient, and differentiable. Facing this challenge required us to make compromises. The projection-based loss function (chapter 2) is not guaranteed to capture all the topological errors, the PH-based loss (chapter 3) captures them stochastically, and running the snake-based loss of chapter 4 and connectivity loss of chapter 2 requires larger computing resources than evaluating the classical loss functions. But we had at least a partial success: Experimental results consistently show that deep networks trained with our loss functions produce delineations with far more correct connectivity than ones trained with local loss functions. Even when the networks make errors, they still produce patterns of thin, connected structures, like the ones shown in Fig 2.8. This seemingly unimportant observation supports our initial hypothesis: to produce predictions with non-local properties, like connectivity, deep networks need to be trained with non-local loss functions. It also shows that deep convolutional networks have the capacity to learn to satisfy non-local properties, if trained adequately. Moreover, it raises questions of scientific relevance that have not been studied so far: given the limited receptive field of ConvNets, what are the limits to their capacity of learning these non-local constraints? Do their designs constrain the space of patterns that can be learned? Are there constraints that they cannot learn to satisfy? Finally, even though the question of designing deep architectures that are suitable for learning connectivity received some attention from the computer vision community, it is still not clear what design aspects make them better in this task.

6.2 Future Work

Downstream tasks requires graphs instead of segmentation maps. Hence the end product of the delineation task is a graph. Most common approach is to predict segmentation maps, then apply thresholding and skeletonization to get graphs. With the proposed methods, we are trying to optimise the topology of the segmentation maps not the end product. Our method presented in chapter 2, make use of spanning trees to find critical pixels that creates topological errors. A promising future research direction would be to use the spanning trees to reconstruct

graphs of the curvilinear structures. We discovered that difference between minimum and maximum spanning trees of a distance map, gives a graph very similar to centerlines. However, there are some unwanted branches that need to be pruned. Furthermore, spanning trees cannot have loops by nature. Hence, networks with loops cannot be fully represented by a tree. We can develop heuristic pruning and merging methods to remove such problems from the spanning tree differences to obtain a proper graph for the centerlines. Most important advantage of this approach is that we will be optimising the topology of the end product thanks to the formulation of the proposed loss.

We already extended this work to a 3D topology-aware loss, however, another way to extend this approach to 3D data, would be using watershed algorithm to find critical voxels instead of spanning trees. By using watershed algorithm on distance maps, we can detect disconnections along the 3D curvilinear structures.

In chapter 3, the proposed approach is limited by the need to randomly select the parameters of the height function at each training iteration, because some orientations of the height axis might result in a failure to detect topological errors, or provoke erroneous matches between the persistence diagrams of the prediction and the ground truth. We plan to tackle this problem by using 4D/5D persistence diagrams. In usual persistence diagrams, there are only two axes: birth and death value. If we also incorporate the spatial coordinates of the critical pixels/voxels into the diagram then the matching of homology classes will be more accurate and it will not be affected by the randomly selected parameters.

Active contour models have an internal energy that is also minimised during the update process. In traditional vision approaches, shape priors of the foreground object can be embedded in this energy. In our approach presented in chapter 4, for all structures, we use a basic internal energy which promotes rigidity and smoothness. A better way to increase convergence accuracy and speed of the contour model on centerlines would be using structure-specific internal energies. We can even let the deep network learn the shape priors since the update of contour model is differentiable. With this method, we can learn shape priors of different curvilinear structures during training and these priors can assist further tasks such as manual tracing with annotation tools. Furthermore, contour models can be projected on feature maps of earlier layers of the deep networks to capture and promote the linear patterns. We can create a curvilinear structure specific pooling operation similar to strided convolution where the weights are learned by the contour model.

In our current implementation of the method proposed in chapter 5, projection direction is used regardless of the delineated structure's shape. Nevertheless, some projections are more insightful than others. To further improve delineation accuracy while reducing the required annotation effort, we can develop algorithms for automatic selection of the optimal projection direction for various parts of the volume, thereby reducing the number of projections to fewer than three and increase the number of captured topological errors.

Bibliography

- [1] Abousamra, S., Hoai, M., Samaras, D., and Chen, C. (2021). Localization in the Crowd with Topological Constraints. In *AAAI Conference on Artificial Intelligence*.
- [2] Acuna, D., Kar, A., and Fidler, S. (2019). Devil is in the Edges: Learning Semantic Boundaries from Noisy Annotations. In *Conference on Computer Vision and Pattern Recognition*.
- [3] Bajcsy, R. and Tavakoli, M. (1976). Computer Recognition of Roads from Satellite Pictures. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(9):623–637.
- [4] Barannikov, S. (1994). The framed Morse complex and its invariants. *Advances in Soviet Mathematics*, 21:93–115.
- [5] Bastani, F. (2018). Roadtracer web page. <https://mapster.csail.mit.edu/roadtracer.html>.
- [6] Bastani, F., He, S., Alizadeh, M., Balakrishnan, H., Madden, S., Chawla, S., Abbar, S., and Dewitt, D. (2018). Roadtracer: Automatic Extraction of Road Networks from Aerial Images. In *Conference on Computer Vision and Pattern Recognition*.
- [7] Batra, A., Singh, S., Pang, G., Basu, S., Jawahar, C., and Paluri, M. (2019). Improved Road Connectivity by Joint Learning of Orientation and Segmentation. In *Conference on Computer Vision and Pattern Recognition*.
- [8] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation Learning: A Review and New Perspectives.
- [9] Betthausen, L. (2018). *Topological reconstruction of grayscale images*. PhD thesis, University of Florida.
- [10] Biagioni, J. and Eriksson, J. (2012). Inferring Road Maps from Global Positioning System Traces: Survey and Comparative Evaluation. *Transportation Research Record: Journal of the Transportation Research Board*, 2291:61–71.
- [11] Bleile, B., Garin, A., Heiss, T., Maggs, K., and Robins, V. (2021). The persistent homology of dual digital image constructions.
- [12] Breitenreicher, D., Sofka, M., Britzen, S., and Zhou, S. (2013). Hierarchical Discriminative Framework for Detecting Tubular Structures in 3D Images. pages 328–340.

Bibliography

- [13] Briggman, K., Denk, W., Seung, S., Helmstaedter, M., and Turaga, S. (2009). Maximin Affinity Learning of Image Segmentation. pages 1865–1873.
- [14] Bullitt, E., Zeng, D., Gerig, G., Aylward, S., Joshi, S., Smith, J., Lin, W., and Ewend, M. (2005). Vessel Tortuosity and Brain Tumor Malignancy: A Blinded Study. *Acad Radiol*, 12(10):1232–1240.
- [15] Butenuth, M. and Heipke, C. (2012). Network Snakes: Graph-Based Object Delineation with Active Contour Models. *Machine Vision and Applications*, 23(1):91–109.
- [16] Byrne, N., Clough, J., Montana, G., and King, A. (2020). A persistent homology-based topological loss function for multi-class CNN segmentation of cardiac MRI. In *STACOM Workshop at MICCAI2020*, volume 12592 of *Lecture Notes in Computer Science*, pages 3–13. Springer.
- [17] Carlsson, G. and Zomorodian, A. (2009). The theory of multidimensional persistence. *Discret. Comput. Geom.*, 42(1):71–93.
- [18] Carrière, M. and Blumberg, A. (2020). Multiparameter persistence image for topological machine learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 22432–22444. Curran Associates, Inc.
- [19] Carriere, M., Chazal, F., Glisse, M., Ike, Y., Kannan, H., and Umeda, Y. (2021). Optimizing persistent homology based functions. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1294–1303. PMLR.
- [20] Chatterjee, S., Prabhu, K., Pattadkal, M., Bortsova, G., Sarasaen, C., Dubost, F., Mattern, H., de Bruijne, M., Speck, O., and Nürnberger, A. (2020). Ds6, Deformation-Aware Semi-Supervised Learning: Application to Small Vessel Segmentation with Noisy Training Data.
- [21] Chaurasia, A. and Culurciello, E. (2017). Linknet: Exploiting Encoder Representations for Efficient Semantic Segmentation. *CoRR*, abs/1707.03718.
- [22] Chen, C., Freedman, D., and Lampert, C. (2011). Enforcing Topological Constraints in Random Field Image Segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 2089–2096.
- [23] Chen, S., Ma, K., and Zheng, Y. (2019). Med3D: Transfer Learning for 3D Medical Image Analysis.
- [24] Cheng, D., Liao, R., Fidler, S., and Urtasun, R. (2019). DARNet: Deep Active Ray Network for Building Segmentation. In *Conference on Computer Vision and Pattern Recognition*.
- [25] Cheng, G., Wang, Y., Xu, S., Wang, H., Xiang, S., and Pan, C. (2017). Automatic Road Detection and Centerline Extraction via Cascaded End-To-End Convolutional Neural Network. *IEEE Trans. Geoscience and Remote Sensing*, 55(6):3322–3337.

-
- [26] Chu, H., Li, D., Acuna, D., Kar, A., Shugrina, M., Wei, X., Liu, M., Torralba, A., and Fidler, S. (2019). Neural Turtle Graphics for Modeling City Road Layouts. In *International Conference on Computer Vision*.
- [27] Clough, J., Byrne, N., Oksuz, I., Zimmer, V., Schnabel, J., and King, A. (2020). A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [28] Clough, J., Oksuz, I., Byrne, N., Schnabel, J., and King, A. (2019). Explicit Topological Priors for Deep-Learning Based Image Segmentation Using Persistent Homology.
- [29] Cootes, T. F., Edwards, G. J., and Taylor, C. J. (2001). Active Appearance Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6).
- [30] Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D., and Raskar, R. (2018). Deepglobe 2018: A Challenge to Parse the Earth through Satellite Images. In *Conference on Computer Vision and Pattern Recognition*.
- [31] Doersch, C. and Zisserman, A. (2017). Multi-Task Self-Supervised Visual Learning.
- [32] Edelsbrunner, H. and Harer, J. (2008). Persistent homology - a survey. *Contemporary mathematics*, 453:257–282.
- [33] Edelsbrunner, H., Letscher, D., and Zomorodian, A. (2000). Topological persistence and simplification. In *Proceedings 41st annual symposium on foundations of computer science*, pages 454–463. IEEE.
- [34] Etten, A. V., Lindenbaum, D., and Bacastow, T. (2018). Spacenet: A Remote Sensing Dataset and Challenge Series.
- [35] Fischler, M., Tenenbaum, J., and Wolf, H. (1981). Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique. *Computer Vision, Graphics, and Image Processing*, 15(3):201–223.
- [36] Frangi, A., Niessen, W., Vincken, K., and Viergever, M. (1998). Multiscale Vessel Enhancement Filtering. *Lecture Notes in Computer Science*, 1496:130–137.
- [37] Fua, P. (1996). Model-Based Optimization: Accurate and Consistent Site Modeling. In *International Society for Photogrammetry and Remote Sensing*.
- [38] Fua, P. and Leclerc, Y. G. (1990). Model Driven Edge Detection. *Machine Vision and Applications*, 3:45–56.
- [39] Funke, J., Tschopp, F. D., Grisaitis, W., Sheridan, A., Singh, C., Saalfeld, S., and Turaga, S. C. (2018). Large Scale Image Segmentation with Structured Loss Based Deep Learning for Connectome Reconstruction. 41(7):1669–1680.

- [40] Gabrielsson, R., Nelson, B., Dwaraknath, A., and Skraba, P. (2020). A topology layer for machine learning. In *International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1553–1563.
- [41] Ganin, Y. and Lempitsky, V. (2014a). N4-Fields: Neural Network Nearest Neighbor Fields. pages 536–551.
- [42] Ganin, Y. and Lempitsky, V. (2014b). N4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms. pages 536–551.
- [43] Garin, A., Heiss, T., Maggs, K. A. R., Bleile, B., and Robins, V. (2020). Duality in persistent homology of images. *ArXiv*.
- [44] Garin, A. and Tauzin, G. (2019). A Topological "Reading" Lesson: Classification of MNIST using TDA. In *International Conference on Machine Learning and Applications*, pages 1551–1556.
- [45] Guo, Z., Bai, J., Lu, Y., Wang, X., Cao, K., Song, Q., Sonka, M., and Yin, Y. (2019). Deepcenterline: A Multi-Task Fully Convolutional Network for Centerline Extraction. In *IPMI*, pages 441–453.
- [46] Hamarneh, G. and Jassi, P. (2010). Vascusynth: Simulating Vascular Trees for Generating Volumetric Image Data with Ground Truth Segmentation and Tree Analysis. *Computerized Medical Imaging and Graphics*, 34(8):605–616.
- [47] Hatamizadeh, A., Sengupta, D., and Terzopoulos, D. (2020). End-To-End Trainable Deep Active Contour Models for Automated Image Segmentation: Delineating Buildings in Aerial Imagery. In *European Conference on Computer Vision*.
- [48] Hu, X., Li, F., Samaras, D., and Chen, C. (2019). Topology-Preserving Deep Image Segmentation. In *Advances in Neural Information Processing Systems*, pages 5658–5669.
- [49] Hu, X., Wang, Y., Fuxin, L., Samaras, D., and Chen, C. (2021). Topology-Aware Segmentation Using Discrete Morse Theory. In *International Conference on Learning Representations*.
- [50] Huang, X. and Zhang, L. (2009). Road Centreline Extraction from High-Resolution Imagery Based on Multiscale Structural Features and Support Vector Machines. *International Journal of Remote Sensing*, 30:1977–1987.
- [51] Ishii, Y., Koizumi, K., Fukami, H., Yamamoto, K., Takahashi, H., Limin, S., Kusin, K., Usup, A., and Susilo, G. (2016). Groundwater in peatland. In *Tropical Peatland Ecosystems*, pages 265–279.
- [52] Jassi, P. and Hamarneh, G. (2011). Vascusynth: Vascular Tree Synthesis Software. *Insight Journal*, January-June:1–12.
- [53] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331.

-
- [54] Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimisation.
- [55] Koziński, M., Mosińska, A., Salzmann, M., and Fua, P. (2018). Learning to Segment 3D Linear Structures Using Only 2D Annotations. pages 283–291.
- [56] Koziński, M., Mosinska, A., Salzmann, M., and Fua, P. (2020). Tracing in 2D to Reduce the Annotation Effort for 3D Deep Delineation of Linear Structures. 60.
- [57] Krissian, K., Malandain, G., Ayache, N., Vaillant, R., and Troussset, Y. (2000). Model-Based Detection of Tubular Structures in 3D Images. 80(1):130–171.
- [58] Law, M. and Chung, A. (2008). Three Dimensional Curvilinear Structure Detection Using Optimally Oriented Flux. pages 368–382.
- [59] Leifeld, J., Wüst-Galley, C., and Page, S. (2019). Intact and managed peatland soils as a source and sink of ghgs from 1850 to 2100. *Nature Climate Change*, 9(12):945–947.
- [60] Leygonie, J., Oudot, S., and Tillmann, U. (2021). A framework for differential calculus on persistence barcodes. *Foundations of Computational Mathematics*.
- [61] Li, X., Wang, Y., Zhang, L., Liu, S., Mei, J., and Li, Y. (2020). Topology-Enhanced Urban Road Extraction via a Geographic Feature-Enhanced Network. *IEEE Trans. Geosci. Remote. Sens.*, 58(12):8819–8830.
- [62] Li, Z., Wegner, J., and Lucchi, A. (2019). Topological Map Extraction from Overhead Images. In *International Conference on Computer Vision*.
- [63] Maninis, K., Pont-Tuset, J., Arbeláez, P., and Gool, L. V. (2016). Deep Retinal Image Understanding. pages 140–148.
- [64] Marcos, D., Tuia, D., Kellenberger, B., and Urtasun, R. (2018). Learning Deep Structured Active Contours End-To-End. In *Conference on Computer Vision and Pattern Recognition*.
- [65] Maria, C., Boissonnat, J. D., Glisse, M., and Yvinec, M. (2014). The gudhi library: Simplicial complexes and persistent homology. In *International congress on mathematical software*, pages 167–174. Springer.
- [66] Mátyus, G., Luo, W., and Urtasun, R. (2017). Deeproadmapper: Extracting Road Topology from Aerial Images. In *International Conference on Computer Vision*, pages 3458–3466.
- [67] Min, S., Chen, X., Zha, Z., Wu, F., and Zhang, Y. (2019). A Two-Stream Mutual Attention Network for Semi-Supervised Biomedical Segmentation with Noisy Labels. pages 4578–4585.
- [68] Mnih, V. (2013). *Machine Learning for Aerial Image Labeling*. PhD thesis, University of Toronto.
- [69] Mnih, V. and Hinton, G. (2010). Learning to Detect Roads in High-Resolution Aerial Images. In *European Conference on Computer Vision*, pages 210–223.

Bibliography

- [70] Mosińska, A., Kozinski, M., and Fua, P. (2020). Joint Segmentation and Path Classification of Curvilinear Structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(6):1515–1521.
- [71] Mosińska, A., Marquez-Neila, P., Kozinski, M., and Fua, P. (2018). Beyond the Pixel-Wise Loss for Topology-Aware Delineation. In *Conference on Computer Vision and Pattern Recognition*, pages 3136–3145.
- [72] Murdiyarso, D., Hergoualc’h, K., and Verchot, L. V. (2010). Opportunities for reducing greenhouse gas emissions in tropical peatlands. *Proceedings of the National Academy of Sciences*, 107(46):19655–19660.
- [73] Oner, D., Citraro, L., Koziński, M., and Fua, P. (2022a). Adjusting the Ground Truth Annotations for Connectivity-Based Learning to Delineate. In *IEEE Transactions on Medical Imaging*.
- [74] Oner, D., Garin, A., Koziński, M., Hess, K., and Fua, P. (2022b). Persistent Homology with Improved Locality Information for more Effective Delineation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [75] Oner, D., Koziński, M., Citraro, L., Dadap, N. C., Konings, A. G., and Fua, P. (2021). Promoting Connectivity of Network-Like Structures by Enforcing Region Separation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- [76] Oner, D., Osman, H., Koziński, M., and Fua, P. (2022c). Enforcing Connectivity of 3D Linear Structures Using Their 2D Projections. In *Conference on Medical Image Computing and Computer Assisted Intervention*.
- [77] Peng, H., Tang, J., Xiao, H., Bria, A., Zhou, J., Butler, V., Zhou, Z., Gonzalez-Bellido, P., Oh, S., and others, C. A. (2014). Virtual Finger Boosts Three-Dimensional Imaging and Microsurgery as Well as Terabyte Volume Image Visualization and Analysis. *Nature Communications*, 5:4342–4355.
- [78] Peng, H., Zhou, Z., E.Meijering, T.Zhao, Ascoli, G., and M.Hawrylycz (2017). Automatic Tracing of Ultra-Volumes of Neuronal Images. *Nature Methods*, 14:332–333.
- [79] Quam, L. (1978). Road Tracking and Anomaly Detection. In *DARPA Image Understanding Workshop*, pages 51–55.
- [80] Ramnath, K., Baker, S., Matthews, I., and Ramanan, D. (2008). Increasing the Density of Active Appearance Models. In *Conference on Computer Vision and Pattern Recognition*.
- [81] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. pages 234–241.
- [82] Seyedhosseini, M., Sajjadi, M., and Tasdizen, T. (2013). Image Segmentation with Cascaded Hierarchical Models and Logistic Disjunctive Normal Networks. In *International Conference on Computer Vision*.

-
- [83] Shit, S., Paetzold, J., Sekuboyina, A., Ezhov, I., Unger, A., Zhylka, A., Pluim, J., Bauer, U., and Menze, B. (2021). cldice - A novel topology-preserving loss function for tubular structure segmentation. In *CVPR*, pages 16560–16569. Computer Vision Foundation / IEEE.
- [84] Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations*.
- [85] Sironi, A., Lepetit, V., and Fua, P. (2014). Multiscale Centerline Detection by Learning a Scale-Space Distance Transform. In *Conference on Computer Vision and Pattern Recognition*.
- [86] Sironi, A., Turetken, E., Lepetit, V., and Fua, P. (2016). Multiscale Centerline Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1327–1341.
- [87] Team, P. (2017). Planet application program interface: In space for life on earth. <https://api.planet.com>.
- [88] Terzopoulos, D., Witkin, A., and Kass, M. (1988). Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion. *Artificial Intelligence*, 36(1):91–123.
- [89] Turetken, E., Becker, C., Glowacki, P., Benmansour, F., and Fua, P. (2013). Detecting Irregular Curvilinear Structures in Gray Scale and Color Imagery Using Multi-Directional Oriented Flux. In *International Conference on Computer Vision*, pages 1553–1560.
- [90] Turetken, E., Benmansour, F., Andres, B., Glowacki, P., Pfister, H., and Fua, P. (2016). Reconstructing Curvilinear Networks Using Path Classifiers and Integer Programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(12):2515–2530.
- [91] Turner, K., Mukherjee, S., and Boyer, D. (2014). Persistent Homology Transform for Modeling Shapes and Surfaces. *Information and Inference*, 3(4):310–344.
- [92] Van Etten, A. (2017). Spacenet Road Detection and Routing Challenge Part II — APLS Implementation.
- [93] Vanderbrug, G. (1976). Line Detection in Satellite Imagery. *IEEE Transactions on Geoscience Electronics*, 14(1):37–44.
- [94] Vernimmen, R., Hooijer, A., Mulyadi, D., Setiawan, I., Pronk, M., and Yuherdha, A. T. (2020). A new method for rapid measurement of canal water table depth using airborne lidar, with application to drained peatlands in indonesia. *Water*, 12(5):1486.
- [95] Wang, C., Li, Y., Ito, W., Shimura, K., and Abke, K. (2009). A Machine Learning Approach to Extract Spinal Column Centerline from Three-Dimensional CT Data.
- [96] Wang, F., Liu, H., Samaras, D., and Chen, C. (2020a). Topogan: A Topology-Aware Generative Adversarial Network. In *European Conference on Computer Vision*, pages 118–136.

Bibliography

- [97] Wang, G., Liu, X., Li, C., Xu, Z., Ruan, J., Zhu, H., Meng, T., Li, K., Huang, N., and Zhang, S. (2020b). A Noise-Robust Framework for Automatic Segmentation of COVID-19 Pneumonia Lesions from CT Images. 39(8):2653–2663.
- [98] Wang, W., Yu, K., Hugonot, J., Fua, P., and Salzmann, M. (2019). Recurrent U-Net for Resource-Constrained Segmentation. In *International Conference on Computer Vision*.
- [99] Wegner, J., Montoya-Zegarra, J., and Schindler, K. (2013). A Higher-Order CRF Model for Road Network Extraction. In *Conference on Computer Vision and Pattern Recognition*, pages 1698–1705.
- [100] Wiedemann, C., Heipke, C., Mayer, H., and Jamet, O. (1998). Empirical Evaluation of Automatically Extracted Road Axes. In *Empirical Evaluation Techniques in Computer Vision*, pages 172–187.
- [101] Wolterink, J., van Hamersvelt, R., Viergever, M., Leiner, T., and Isgum, I. (2019). Coronary Artery Centerline Extraction in Cardiac CT Angiography Using a Cnn-Based Orientation Classifier. *Medical Image Anal.*, 51:46–60.
- [102] Wu, D., Reisinger, D., Xu, J., Fatemi, S., van Zijl, P., Mori, S., and Zhang, J. (2014). Localized diffusion magnetic resonance micro-imaging of the live mouse brain. *NeuroImage*, 91:12–20.
- [103] Yang, X., Li, X., Ye, Y., Lau, R. Y. K., Zhang, X., and Huang, X. (2019). Road Detection and Centerline Extraction via Deep Recurrent Convolutional Neural Network U-Net. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–12.
- [104] Yu, Z., Liu, W., Zou, Y., Feng, C., Ramalingam, S., Vijaya, K., and Kautz, J. (2018). Simultaneous Edge Alignment and Learning. In *European Conference on Computer Vision*.
- [105] Zhang, Z., Marin, D., Chesakov, E., Maza, M. M., Drangova, M., and Boykov, Y. (2019). Divergence Prior and Vessel-Tree Reconstruction. pages 10216–10224.
- [106] Zhou, Z., Liu, X., Long, B., and Peng, H. (2016). TReMAP: Automatic 3D Neuron Reconstruction Based on Tracing, Reverse Mapping and Assembling of 2D Projections. *Neuroinformatics*, 14(1):41–50.
- [107] Zhu, H., Shi, J., and Wu, J. (2019). Pick-And-Learn: Automatic Quality Evaluation for Noisy-Labeled Image Segmentation. pages 576–584.
- [108] Zomorodian, A. and Carlsson, G. (2004). Computing persistent homology. In *ACM Symposium on Computational Geometry*, pages 347–356. ACM.

Doruk Oner

DOCTORAL ASSISTANT

☎ (+41) 78 345 78 48 | ✉ doruk.oner@epfl.ch | 🏠 doruk-oner.github.io | 📷 doruk-oner | 🌐 doruk-oner

Education

EPFL - Ecole Polytechnique Federale de Lausanne

PHD CANDIDATE IN COMPUTER SCIENCE

Lausanne, Switzerland

2018 - (Expected June 2023)

- Computer Vision Laboratory (CVLAB), Supervisor: Prof. Pascal Fua
- EDIC Fellowship

Bilkent University

B.S. IN ELECTRICAL AND ELECTRONICS ENGINEERING

Ankara, Turkey

2014 - 2018

- CGPA: 3.90/4.00 (High Honor Student)
- Comprehensive Scholarship

Honors & Awards

2018	EPFL EDIC Fellowship
2018	Bilkent University Academic Excellence Award
2014 - 2018	Bilkent University Full/Comprehensive Scholarship
2014 - 2018	Bilkent University High Honour Student
2014	Ranked 435th in the National University Entrance Exam among 2.086.087 students
2010 - 2014	TED Ankara College Full/Comprehensive Scholarship
2012	Qualified to Central Anatolia Region Finals in Physics Branch of TUBITAK Science Project Competition
2010 - 2014	Member of TUBITAK National Mathematics Olympiads Team in TED Ankara College
2011	Qualified to 2nd level examination of TUBITAK National Mathematics Olympiads
2011	Qualified to Turkey Finals in International Geography Olympics

Publications

TPAMI - 2023	D. Oner , A. Garin, M. Koziński, K. Hess, P. Fua Persistent Homology with Improved Locality Information for more Effective Delineation
MICCAI - 2022	D. Oner , Hussein Osman, M. Koziński, P. Fua Enforcing connectivity of 3D linear structures using their 2D projections
TMI - 2022	D. Oner , L. Citraro, M. Koziński, P. Fua Adjusting the Ground Truth Annotations for Connectivity-Based Learning to Delineate
TPAMI 2021	D. Oner , M. Koziński, L. Citraro, N.C. Dadap, A.G. Konings, P. Fua Promoting Connectivity of Network-Like Structures by Enforcing Region Separation
Agu Advances 2021	N.C. Dadap, A.M. Hoyt, A.R. Cobb, D. Oner , M. Koziński, P. Fua, K. Rao, C.F. Harvey, A.G. Konings Drainage Canals in Southeast Asian Peatlands Increase Carbon Emissions
ConBuildMat 2021	B.G. Pantoja-Rosero, D. Oner , M. Koziński, R. Achanta, P. Fua, F. Perez-Cruz, K. Beyer TOPO-Loss for Continuity-Preserving Crack Detection Using Deep Learning
AISTATS 2018	E. Turgay, D. Oner , C. Tekin Multi-Objective Contextual Bandit Problem with Similarity Information
preprint 2018	D. Oner , A. Karakurt, A. Eryilmaz, C. Tekin Combinatorial Multi-Objective Multi-Armed Bandit Problem

Skills

Machine Learning	PyTorch, Scikit-learn, Keras
Visual Computing	Numpy, SciPy, OpenCV
Programming	Python, C++, MATLAB, VHDL, Assembly
Other	Git, Latex, AWS, Kubernetes, Sagemaker
Languages	English (Very fluent, TOEFL score: 110/120), Turkish (Native), French (Beginner)

103

Experience

ARGEDOR

PART-TIME MACHINE LEARNING ENGINEER

Ankara, Turkey

Aug. 2017 - Aug. 2018

- Ensemble of deep learning methods (Convolutional neural networks and LSTM) and decision trees (Light GBM and XGBoost) are used to rank altcoins and Forex parities to find most profitable trading strategies.
- Creating simulations tools to evaluate the developed methods.
- Python, Keras, Tensorflow, Light GBM, XGBoost

ARGEDOR

INTERN

Ankara, Turkey

Jul. 2017 - Aug. 2017

- Convolutional neural networks are used to find optimal Forex trading strategies.
- Python, Keras, Tensorflow

Teaching and Services

Courses

- Teaching assistant for "General Physics: Mechanics" course.
- Teaching assistant for "Probabilities and Statistics" course.
- Teaching assistant for "Information, Computation, Communication" course.

Students

- Supervised master's thesis of Raphaël Mariétan.
- Supervised semester projects of Hussein Osman, Taha Zakariya, Michele Pettinato, Taha Mounir, and Jakub Jan Gwizdala.

Reviewing

- Reviewer for CVPR.
- Reviewer for MICCAI.
- Reviewer for TMI.

Extracurricular Activity

- Licensed Player of METU Tennis Team for 9 years.
- Licensed Catamaran sailor.
- Innosuisse - Business Creation in Engineering Entrepreneurship Certificate.
- Innosuisse - Business Concept Entrepreneurship Certificate.
- Landscape and Portrait Photography.
- Organization Team Member and Presenter in YÖNET 2015, a leadership camp with 200 participants.
- Organization Team Member in CASE CHALLENGE 2015, a case study competition.