Learning Topological Representations for Deep Image Understanding

A Dissertation presented

by

Xiaoling Hu

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2023

Stony Brook University

The Graduate School

Xiaoling Hu

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Chao Chen, Advisor
Assistant Professor, Department of Biomedical Informatics

Dimitris Samaras, Committee Member
SUNY Empire Innovation Professor, Department of Computer Science

Haibin Ling, Committee Member
SUNY Empire Innovation Professor, Department of Computer Science

Fuxin Li, Outside Committee Member
Associate Professor, Department of EECS, Oregon State University

This dissertation is accepted by the Graduate School

Celia Marshik
Dean of the Graduate School

Abstract of the Dissertation

Learning Topological Representations for Deep Image Understanding

by

Xiaoling Hu

Doctor of Philosophy

in

Computer Science

Stony Brook University

2023

In many scenarios, especially biomedical applications, the correct delineation of complex fine-scaled structures such as neurons, tissues, and vessels is critical for downstream analysis. Despite the strong predictive power of deep learning methods, they do not provide a satisfactory representation of these structures, thus creating significant barriers in scalable annotation and downstream analysis. In this dissertation, we tackle such challenges by proposing novel representations of these topological structures in a deep learning framework. We leverage the mathematical tools from topological data analysis, i.e., persistent homology and discrete Morse theory, to develop principled methods for better segmentation and uncertainty estimation, which will become powerful tools for scalable annotation.

We focus on a few specific problems. First, we propose novel topological losses for fully-supervised segmentation. Although deep-learning-based segmentation methods have achieved satisfactory segmentation performance in terms of per-pixel accuracy, most of them are still prone to structural errors, e.g., broken connections and missing connected components. We propose topological losses to teach neural networks to segment with correct topology. The continuous-valued loss functions enforce a segmentation to have a better topology by penalizing topologically critical pixels/locations. Second, we focus on the interactive setting, where uncertainty measurement of a neural network segmenter is crucial for scalable annotation. Ex-

isting methods only learn pixel-wise feature representations. We move from pixel space to structure space using the classic discrete Morse theory. We decompose an input image into structural elements such as branches and patches and then learn a probabilistic model over such structural space. Our method effectively identifies hypothetical structures that a model is uncertain about and asks a domain expert to confirm. This will significantly improve the annotation speed.

Dedicated to my family.

Contents

List of Figures

List of Tables

Acknowledgements

This thesis would not have been possible without the invaluable support of my mentors, friends, and family.

First and foremost, I am indebted to my advisor, Prof. Chao Chen. I consider myself incredibly fortunate to have had him as my guide throughout this journey. His qualities as a stellar advisor are unparalleled: his passion for research, profound knowledge in the field, unwavering dedication to his work, keen attention to details, and constant advocacy for my best interests. Over the years, he has steadfastly navigated me through both the peaks and troughs of research and life, imparting a consistent and effective set of principles. Chao Chen has, without a doubt, been the epitome of an ideal advisor - a dream come true. His mentorship will forever remain an inspiration and guiding light throughout my life.

Next, I extend my heartfelt gratitude to my esteemed committee members, Prof. Dimitris Samaras, Prof. Haibin Ling, and Prof. Fuxin Li, for their valuable contributions to my dissertation.

Throughout my entire Ph.D. journey, I am profoundly grateful to my colleagues, collaborators, and friends for engaging in enriching discussions about my work and career. I would like to extend my heartfelt appreciation to the following individuals, among many others, who have played pivotal roles in supporting me: Shahira Abousamra, Saumya Gupta, Wentao Huang, Aishik Konwer, Chen Li, Prateek Prasanna, Yusu Wang, Meilong Xu, Jiaqi Yang, Jiachen Yao.

Finally, my deepest thanks go to my parents. Your unconditional love and unwavering support have been the bedrock of my achievements, and I acknowledge that I would not have reached this milestone without you.

Vita

Xiaoling Hu was born in Hubei Province, China. He received his B.S. degree from Huazhong University of Science and China in 2014, and M.S. degree from Tsinghua University in 2017. In January 2018, he began his PhD studies in the Department of Computer Science at Stony Brook University. Xiaoling's research interest lies in the intersection of medical imaging, computer vision, and machine learning. His goal is to build AI systems that can efficiently assist in disease diagnosis and treatment. In particular, he strives to create novel algorithms with both concrete theoretical foundations and strong empirical results for imaging data under different contexts, especially biomedical scenarios.

Chapter 1

Introduction and Overview

In many scenarios, especially biomedical applications, the correct delineation of complex fine-scaled structures such as neurons, tissues, and vessels is critical for downstream analysis. Despite the strong predictive power of deep learning methods, they are only learning pixel-wise representations, thus creating significant barriers in scalable annotation and downstream analysis.

The goal of this dissertation is to explore beyond pixel-wise representations: *How can we be able to learn topological representations to understand the topology/structures for image tasks*?

In this dissertation, we tackle such challenges by proposing novel representations of these topological structures in a deep learning framework. We leverage the mathematical tools from topological data analysis, i.e., persistent homology and discrete Morse theory, to develop principled methods for better segmentation and uncertainty estimation, which will become powerful tools for scalable annotation.

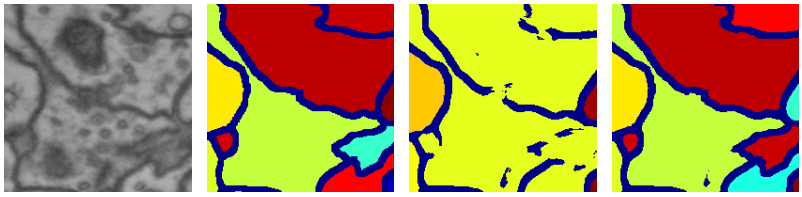
1.1 Topological Loss for Image Segmentation

Image segmentation, i.e., assigning labels to all pixels of an input image, is crucial in many computer vision tasks. State-of-the-art deep segmentation methods [10–14] learn high quality feature representations through an end-to-end trained deep network and achieve satisfactory per-pixel accuracy.

However, these segmentation algorithms are still prone to make errors on fine-scaled structures, such as small object instances, instances with multiple connected components, and thin connections. These fine-scaled structures may be crucial in analyzing the *functionality* of the objects. For example, accurate extraction of thin

parts such as ropes and handles is crucial in planning robot actions, e.g., dragging or grasping. In biomedical images, correct delineation of thin objects such as neuron membranes and vessels is crucial in providing accurate morphological and structural quantification of the underlying system. A broken connection or a missing component may only induce marginal per-pixel error but can cause catastrophic functional mistakes.

One example is neuron image segmentation. When we do segmentation for neuron images, the goal of this task is to segment membranes that partition the image into regions corresponding to neurons. As illustrated in Fig. 1.1(c), though the method achieves good enough performance in terms of pixel accuracy, it makes errors at some connection parts, resulting in the wrong partition of some regions. These wrong partitions will definitely affect the downstream functionality analysis of the neurons.



(a) Original (b) GT (c) DIVE [1] (d) Ours

Figure 1.1: Illustration of the importance of topological correctness in a neuron image segmentation task. The goal of this task is to segment membranes that partition the image into regions corresponding to neurons. **(a)** an input neuron image. **(b)** ground truth segmentation of the membranes (dark blue) and the result neuron regions. **(c)** result of a baseline method without topological guarantee [1]. Small pixel-wise errors lead to broken membranes, resulting in the merging of many neurons into one. **(d)** Our method produces the correct topology and the correct partitioning of neurons [2].

At the initial step, with the help of topological data analysis, a novel method based on persistent homology is proposed to tackle structural accuracy directly. The proposed topology-preserving loss function is differentiable and is easy to be incorporated into deep neural networks. Also, we extend the proposed method to 3D cases. The proposed method achieves much better performance in terms of several topology-relevant metrics, e.g., the Adjusted Rand Index and the Variation of Information. We conduct experiments on several natural and medical image datasets. We discuss the approach and present our results for this work in Chapter 3.

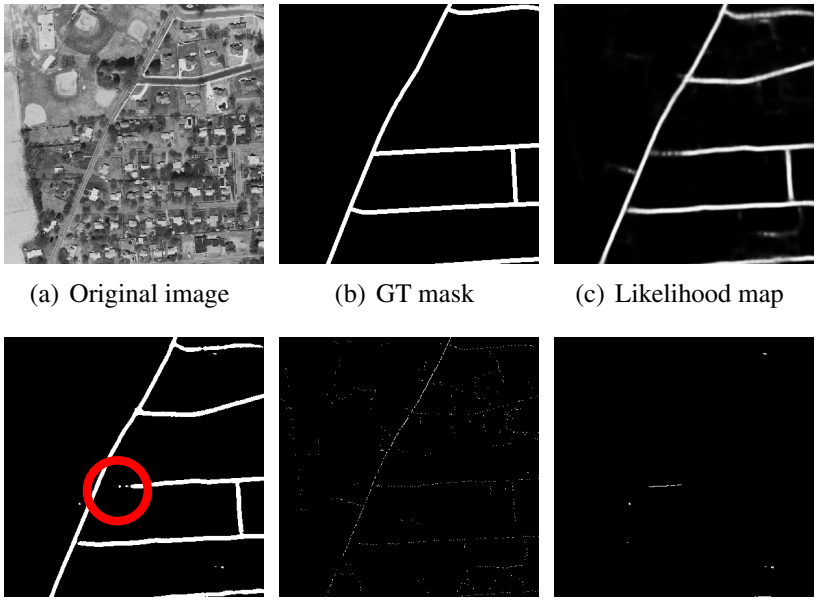
Persistent homology is a powerful tool to analyze the geometric properties of low dimensional data, including images, which can be regarded as 2-dimensional data. Encouraged by the promising performances on the segmentation tasks, we further explore the power of the topological loss in a quite different context, i.e., trojan detection. More specifically, we propose a novel target-label-agnostic reverse engineering method. First, to improve the quality of the recovered triggers, we need a prior that can localize the triggers, but in a flexible manner. We, therefore, propose to enforce a *topological prior* to the optimization process of reverse engineering, i.e., the recovered trigger should have fewer connected components. This prior is implemented through a topological loss introduced in Chapter 3. It allows the recovered trigger to have arbitrary shape and size. Meanwhile, it ensures the trigger is not scattered and is reasonably localized. We discuss the approach and present our results for this work in Chapter 4.

Though TopoLoss achieves quite good performances, the identified critical points can be very noisy and often are not relevant to topological errors. See Fig. 1.2 as an illustration. Moreover, the computation of persistent homology is expensive, making it difficult to evaluate the loss and gradient at every training iteration. Compared to the proposed method (Fig. 1.2(f)), the critical points identified from [2] are very noisy and often are not relevant to the topological errors.

Instead, we propose a novel *homotopy warping loss*, which penalizes errors on topologically critical pixels. These locations are defined by homotopic warping of predicted and ground truth masks. The loss can be incorporated into the training of topology-preserving deep segmentation networks. We discuss the approach and present our results for this work in Chapter 5.

1.2 Beyond Pixel-Wise Representation

In both Chapter 3 and Chapter 5, we introduce methods identifying a set of critical points of the likelihood function, e.g., saddles and extrema, as topologically critical locations for the neural network to memorize. However, only identifying a *sparse set* of critical points at every epoch is inefficient in terms of training. Instead, our method identifies a much bigger set of critical locations at each epoch, i.e., 1D or 2D Morse skeletons (curves and patches). This is beneficial in both training efficiency and model performance. Extending the critical location sets from points to 1D curves and 2D patches makes it much more efficient in training. Furthermore, by focusing on more critical locations early, our method is more likely to escape



(a) Original image (b) GT mask (c) Likelihood map

(d) Thresholded mask (e) TopoNet (f) Warping

Figure 1.2: Illustration of the critical points identified by different methods. **(a)**: Original image. **(b)**: GT mask. **(c)**: Predicted likelihood map. **(d)**: Segmentation (Thresholded mask from likelihood map). **(e)**: Critical points identified by [2]. **(f)**: Critical points identified by our homotopy warping. Please zoom-in for better viewing.

poor local minima of the loss landscape. Thus it achieves better topological accuracy than TopoLoss. The shorter training time may also contribute to better stability of the SGD algorithm, and thus better test accuracy [15].

However, these loss-based methods are still not ideal. They are based on a standard segmentation network, and thus *only learn pixel-wise feature representations*. This causes several issues. First, a standard segmentation network makes pixel-wise predictions. Thus, at the inference stage, topological errors, e.g. broken connections, can still happen, even though they may be mitigated by the topology-inspired losses. Another issue is in uncertainty estimation, i.e., estimating how certain a segmentation network is at different locations. Uncertainty maps can direct the focus of human annotators for efficient proofreading. However, for fine-scale structures, the existing pixel-wise uncertainty map is not effective.

To fundamentally address these issues, we propose to directly model and

reason about the structures. In this paper, we propose *the first deep learning based method that directly learns the topological/structural representation of images*. To move from pixel space to structure space, we apply the classic discrete Morse theory [16–18] to decompose an image into a Morse complex, consisting of structural elements like branches, patches, etc. These structural elements are hypothetical structures one can infer from the input image. Their combinations constitute a space of structures arising from the input image. We discuss the approach and present our results for this work in Chapter 7.

1.3 Outline

The rest of this dissertation is organized as follows: In Chapter 2, we review different works related to the research of this dissertation. In Chapter 3, Chapter 5 and Chapter 6, we introduce different methods to learn to segment with correct topology. In Chapter 4, we explore the application of topological prior in the trojan detection context. In Chapter 7, we introduce the first deep learning based method that directly learns the topological/structural representation of images. Chapter 8 is the conclusion and future work.

Chapter 2

Related Work

In this chapter, we review the existing works relevant to our research which will be presented in Chapters 3 through 7.

2.1 Topological Data Analysis and Persistent Homology

Topological data analysis (TDA) is a field in which one analyzes datasets using topological tools such as persistent homology [19–25]. The theory has been applied to different applications [2, 8, 26–38].

With the advent of deep learning, some works have tried to incorporate topological information into deep neural networks, and the differentiable property of persistent homology makes it possible. The main idea is that the persistence diagram/barcodes can capture all the topological changes, and it is differentiable to the original data. [2] first propose a topological loss to learn to segment images with correct topology, by matching persistence diagrams in a supervised manner. Similarly, [39] use the persistence barcodes to enforce a given topological prior to the target object. These methods achieve better results, especially in structural accuracy. Persistent-homology-based losses have been applied to other imaging [40, 41] and learning problems [42–45].

The aforementioned methods use topological priors in supervised learning tasks (namely, segmentation). Instead, in this work, we propose to leverage the topological prior in an unsupervised setting; we use a topological prior for the reverse engineering pipeline to reduce the search space of triggers and enforce the recovered triggers to have fewer connected components.

2.2 Attention Mechanism

Attention modules model relationships between pixels/channels/feature maps and have been widely applied in both vision and natural language processing tasks [46–48]. Specifically, the self-attention mechanism [49] is proposed to draw global dependencies of inputs and has been used in machine translation tasks. [50] tries to learn a better image generator via a self-attention mechanism. [51] mainly explores the effectiveness of non-local operation, which is similar to the self-attention mechanism. [52] learns an attention map to aggregate contextual information for each individual point for scene parsing.

2.3 Deep Image Segmentation

Deep learning methods (CNNs) have achieved satisfying performances for image segmentation [6, 10, 12–14, 53–78]. By replacing fully connected layer with fully convolutional layers, FCN [10] transforms a classification CNNs (e.g., AlexNet [79, 80], VGG [81], or ResNet [82]) to fully-convolutional neural networks. In this way, FCN successfully transfers the success of image classification [79, 81, 83] to dense prediction/image segmentation. Instead of using Conditional Random Field (CRF) as post-processing, Deeplab (v1-v2) [12, 13] methods add another fully connected CRF after the last CNN layer to make use of global information. Moreover, Deeplab v3 [14] introduces dilated/atrous convolution to increase the receptive field and make better use of context information to achieve better performance.

Besides the methods mentioned above, UNet [6] has also been one of the most popular methods for image segmentation, especially for images with fine structures. UNet architecture is based on FCN with two major modifications: 1) Similar to the encoder-decoder, UNet is symmetric. The output is the same size as input images, thus suitable for dense prediction/image segmentation, and 2) Skip connections between downsampling and upsampling paths. The skip connections of UNet are able to combine low level/local information with high level/global information, resulting in better segmentation performance.

We also note that many deep learning techniques have been proposed to ensure the segmentation output preserves details, and thus preserves topology implicitly [84–87].

Though obtaining satisfying pixel performances, these methods are still prone to structural/topological errors, as they are usually optimized via pixel-wise loss functions, such as mean-square-error loss (MSE) and cross-entropy loss.

2.4 Topology-Aware Deep Image Segmentation

Beyond pixel-wise accuracy, researchers have tried to segment with the correct shape/topology/geometry [88–93].

One of the closest methods to ours is by Mosinska *et al.* [94], which also proposes a topology-aware loss. Instead of actually computing and comparing the topology, their approach uses the response of selected filters from a pretrained VGG19 network to construct the loss. These filters prefer elongated shapes and thus alleviate the broken connection issue. But this method is hard to generalize to more complex settings with connections of arbitrary shapes. Furthermore, even if this method achieves zero loss, its segmentation is not guaranteed to be topologically correct.

Another closely related to our method is recent works on persistent-homology-based losses [39]. These methods identify a set of critical points of the likelihood function, e.g., saddles and extrema, as topologically critical locations for the neural network to memorize. However, only identifying a *sparse set* of critical points at every epoch is inefficient in terms of training. Instead, our method identifies a much bigger set of critical locations at each epoch, i.e., 1D or 2D Morse skeletons (curves and patches). This is beneficial in both training efficiency and model performance. Extending the critical location sets from points to 1D curves and 2D patches makes it much more efficient in training.

Different ideas have been proposed to capture fine details of objects, mostly revolving around deconvolution and upsampling [6, 10, 12–14, 53]. However, these methods focus on the prediction accuracy of individual pixels and are intrinsically topology-agnostic. Topological constraints, e.g., connectivity and loop-freeness, have been incorporated into variational [26, 95–99] and MRF/CRF-based segmentation methods [100–107]. However, these methods focus on enforcing topological constraints in the inference stage, while the trained model is agnostic of the topological prior. In neuron image segmentation, some methods [55, 108] directly find an optimal partition of the image into neurons and thus avoid segmenting membranes. These methods cannot be generalized to other structures, e.g., vessels, cracks, and roads.

Other methods indirectly preserve topology by enhancing the curvilinear structures. Several methods extract the skeletons of the masks and penalize heavily on pixels of the skeletons. This ensures the prediction is correct along the skeletons and thus is likely correct in topology. clDice [109] extracts the skeleton through min/max-pooling operations over the likelihood map.

One may also use topological constraints as postprocessing steps once we have

the predicted likelihood maps [26, 95–107]. Compared to end-to-end methods, postprocessing methods usually contain self-defined parameters or hand-crafted features, making it difficult to generalize to different situations.

For completeness, we also refer to other existing works on topological features and their applications [26, 30, 31, 110–115]. In graphics, the topological similarity was used to simplify and align shapes [116]. Chen et al. [45] proposed a topological regularizer to simplify the decision boundary of a classifier. Deep neural networks have also been proposed to learn from topological features directly extracted from data [24, 117]. As for deep neural networks, Hofer *et al.* [117] proposed a CNN-based topological classifier. This method directly extracts topological information from an input image/shape/graph as input for CNN, hence cannot generate segmentations that preserve topological priors learned from the training set.

Persistent-homology-inspired objective functions have been proposed for graphics [116] machine learning [42,45]. Discrete Morse theory has been used to identify skeleton structures from images; e.g., [118–120]. The resulting 1D Morse structure has been used to enhance neural network architecture: e.g., in [121] it is used for both pre- and post-process images, while in [122], the 1D Morse skeleton is used as a topological prior (part of the input) for an encoder-decoder deep network for semantic segmentation of microscopic neuroanatomical data. Our work, in contrast, uses Morse structures (beyond 1D) of the output to strengthen the global structural signal more explicitly in an end-to-end training of a network.

Specific to neuron image segmentation, some methods [55, 108, 123–125] directly find neuron regions instead of their boundary/membranes. These methods cannot be generalized to other types of data such as satellite images, retinal images, vessel images, etc.

Additionally, the discrete Morse complex has been used for image analysis, but only as a preprocessing step [118–121] or as a conditional input of a neural network [122].

2.5 Trojan Detection

Many Trojan detection methods [126–134] have been proposed recently. Some focus on detecting poisoned inputs via anomaly detection [135–138]. For example, SentiNet [135] tries to identify adversarial inputs and uses the behaviors of these adversarial inputs to detect Trojaned models. Others focus on analyzing the behaviors of the trained models [5, 139–141]. Specifically, [139] propose the Activation

Clustering (AC) methodology to analyze the activations of neural networks to determine if a model has been poisoned or not.

While early works require all training data to detect Trojans [136, 139, 142], recent approaches have been focusing on a more realistic setting – when one has limited access to the training data. A particularly promising direction is reverse engineering approaches, which recover Trojan triggers with only a few clean samples. Neural cleanse (NC) [3] develops a Trojan detection method by identifying if there is a trigger that would produce misclassified results when added to an input. However, as pointed out by [5], NC becomes futile when triggers vary in terms of size, shape, and location.

Since NC, different approaches have been proposed, extending the reverse engineering idea. Using a conditional generative model, DeepInspect [143] learns the probability distribution of potential triggers from a model of interest. [144] propose to learn universal patterns that change predictions of the model (called Universal Litmus Patterns (ULPs)). The method is efficient as it only involves forward passes through a CNN and avoids backpropagation. ABS [4] analyzes inner neuron behaviors by measuring how extra stimulation can change the network's prediction. [145] propose a data-limited TrojanNet detector (TND) by comparing the impact of the per-sample attack and universal attack. [5] cast Trojan detection as a non-convex optimization problem and it is solved by optimizing an objective function. [146] solve the problem by observing that, compared with clean models, adversarial perturbations transfer from image to image more readily in poisoned models. [147] inspect neural network structure using persistent homology and identify structural cues differentiating Trojaned and clean models.

Existing methods are generally demanding training data access, neural network architectures, types of triggers, target class, etc. This limits their deployment to real-world applications. As for reverse engineering approaches, it remains challenging, if not entirely infeasible, to recover the true triggers. We propose a novel reverse engineering approach that can recover the triggers with high quality using the novel diversity and topological prior. Our method shares the common benefit of reverse engineering methods; it only needs a few clean input images per model. Meanwhile, our approach is agnostic of model architectures, trigger types, and target labels.

2.6 Segmentation Uncertainty

Uncertainty estimation has been the focus of research in recent years [148–155]. However, most existing work focuses on the classification problem. In terms of image segmentation, the research is still relatively limited. Some existing methods directly apply classification uncertainty to individual pixels, e.g., dropout [156,157]. This, however, is not taking into consideration the image structures. Several methods estimate the uncertainty by generating an ensemble of segmentation [150] or using multi-heads [158, 159]. Notably, Probabilistic-UNet [7] learns a distribution over the latent space and then samples over the latent space to produce segmentation samples. When it comes down to uncertainty, however, these methods can still only generate a pixel-wise uncertainty map, using the frequency of appearance of each pixel in the sample segmentations. These methods are fundamentally different from ours, which makes predictions on the structures.

Chapter 3

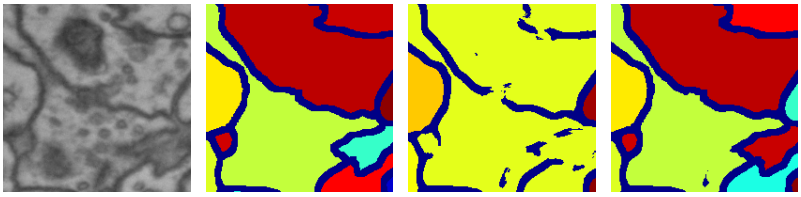
Topology-Preserving Deep Image Segmentation via TopoLoss

In this chapter, we introduce our first try to learn to segment with correct topology by leveraging the theory of persistent homology. We propose a differentiable loss that can be incorporated into any segmentation backbones to improve the topology-aware segmentation accuracy.

3.1 Introduction

Image segmentation, i.e., assigning labels to all pixels of an input image, is crucial in many computer vision tasks. State-of-the-art deep segmentation methods [10–14] learn high quality feature representations through an end-to-end trained deep network and achieve satisfactory per-pixel accuracy. However, these segmentation algorithms are still prone to make errors on fine-scale structures, such as small object instances, instances with multiple connected components, and thin connections. These fine-scale structures may be crucial in analyzing the *functionality* of the objects. For example, accurate extraction of thin parts such as ropes and handles is crucial in planning robot actions, e.g., dragging or grasping. In biomedical images, correct delineation of thin objects such as neuron membranes and vessels is crucial in providing accurate morphological and structural quantification of the underlying system. A broken connection or a missing component may only induce marginal per-pixel error but can cause catastrophic functional mistakes. See Fig. 3.1 for an example.

We propose a novel deep segmentation method that *learns to segment with*



(a) Original (b) GT (c) DIVE [1] (d) Ours

Figure 3.1: Illustration of the importance of topological correctness in a neuron image segmentation task. The goal of this task is to segment membranes that partition the image into regions corresponding to neurons. **(a)** an input neuron image. **(b)** ground truth segmentation of the membranes (dark blue) and the result neuron regions. **(c)** result of a baseline method without topological guarantee [1]. Small pixel-wise errors lead to broken membranes, resulting in the merging of many neurons into one. **(d)** Our method produces the correct topology and the correct partitioning of neurons.

correct topology. In particular, we propose a *topological loss* that enforces the segmentation results to have the same topology as the ground truth, i.e., having the same *Betti number* (number of connected components and handles). A neural network trained with such loss will achieve high topological fidelity without sacrificing per-pixel accuracy. The main challenge in designing such loss is that topological information, namely, Betti numbers, are discrete values. We need a continuous-valued measurement of the topological similarity between a prediction and the ground truth, and such measurement needs to be differentiable in order to backpropagate through the network.

To this end, we propose to use theory from computational topology [19], which summarizes the topological information from a continuous-valued function (in our case, the likelihood function f is predicted by a neural network). Instead of acquiring the segmentation by thresholding f at 0.5 and inspecting its topology, *persistent homology* [19, 20, 160] captures topological information carried by f over all possible thresholds. This provides a unified, differentiable approach of measuring the topological similarity between f and the ground truth, called the *topological loss*. We derive the gradient of the loss so that the network predicting f can be optimized accordingly. We focus on 1-dimensional topology (connections) in 2-dimensional images. And we will focus on 2-dimensional topology (number of voids) when dealing with 3-dimensional images.

Our method is the first end-to-end deep segmentation network with guaranteed

topological correctness. We show that when the topological loss is decreased to zero, the segmentation is guaranteed to be topologically correct, i.e., have identical topology as the ground truth. Our method is empirically validated by comparing with state-of-the-art on natural and biomedical datasets with fine-scaled structures. It achieves superior performance on metrics that encourage structural accuracy. In particular, our method significantly outperforms others on the Betti number error which exactly measures the topological accuracy. Fig. 3.1 shows a qualitative result.

Our method shows how topological computation and deep learning can be mutually beneficial. While our method empowers deep nets with advanced topological constraints, it is also a powerful approach to topological analysis; the observed function is now learned with a highly nonlinear deep network. This enables topology to be estimated based on a semantically informed and denoised observation.

3.2 Method

Our method achieves both per-pixel accuracy and topological correctness by training a deep neural network with a new topological loss, $L_{topo}(f, g)$. Here f is the likelihood map predicted by the network and g is the ground truth. The loss function on each training image is a weighted sum of the per-pixel cross-entropy loss, L_{bce}, and the topological loss:

$$L(f, g) = L_{bce}(f, g) + \lambda L_{topo}(f, g), \tag{3.1}$$

in which λ controls the weight of the topological loss. We assume a binary segmentation task. Thus, there is one single likelihood function f, whose value ranges between 0 and 1.

In Sec. 3.2.1, we introduce the mathematical foundation of topology and how to measure the topology of a likelihood map robustly using persistent homology. In Sec. 3.2.2, we formalize the topological loss as the difference between the persistent homology of f and g. We derive the gradient of the loss and prove its correctness. In Sec. 3.2.3 we explain how to incorporate the loss into the training of a neural network. Although we fix one architecture in experiments, our method is general and can use any neural network that provides pixel-wise prediction. Fig. 3.2 illustrates the overview of our method.

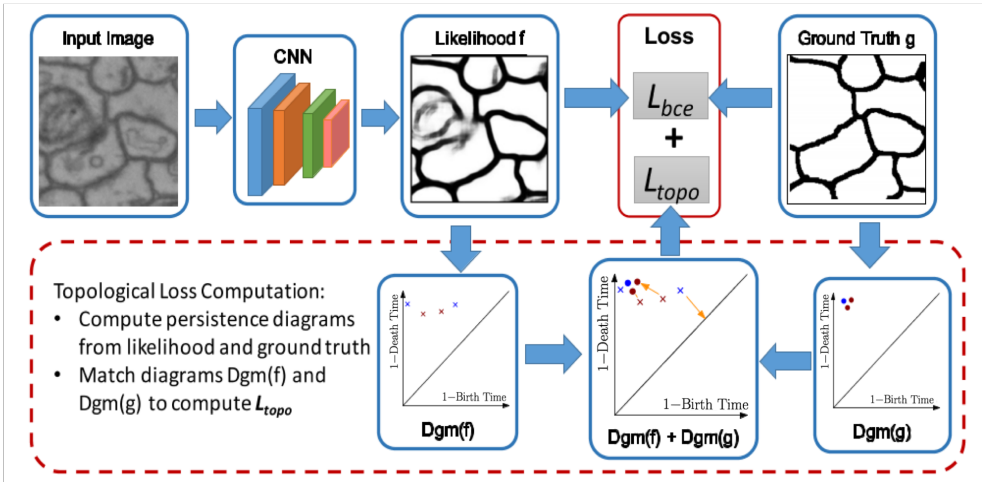


Figure 3.2: An overview of our method.

3.2.1 Topology and Persistent Homology

Given a continuous image domain, $\Omega \subseteq \mathbb{R}^2$ (e.g., a 2D rectangle), we study a likelihood map $f(x) : \Omega \to \mathbb{R}$, which is predicted by a deep neural network (Fig. 3.3(c)). [1] Note that in practice, we only have samples of f at all pixels. In such a case, we extend f to the whole image domain Ω by linear interpolation. Therefore, f is piecewise-linear and is controlled by values at all pixels. A segmentation, $X \subseteq \Omega$ (Fig. 3.3(a)), is calculated by thresholding f at a given value α (often set to 0.5).

Given X, its d-dimension topological structure called a *homology class* [19, 161], is an equivalence class of d-manifolds which can be deformed into each other within X. [2] In particular, 0-dim and 1-dim structures are connected components and handles, respectively. For example, in Fig. 3.3(a), the segmentation X has two connected components and one handle. Meanwhile, the ground truth (Fig. 3.3(b)) has one connected component and two handles. Given X, we can compute the number of topological structures, called the *Betti number*, and compare it with the topology of the ground truth.

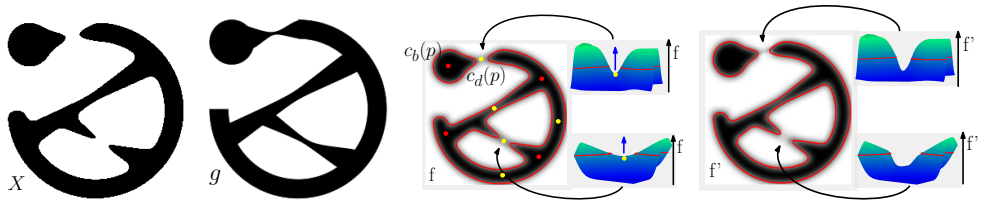
However, simply comparing Betti numbers of X and g will result in a discrete-valued topological error function. To incorporate topological prior into deep neural networks, we need a continuous-valued function that can reveal subtle differences

[1] f depends on the network parameter ω, which will be optimized during training. For convenience, we only use x as the argument of f.

[2] To be exact, a homology class is an equivalent class of cycles whose difference is the boundary of a $(d+1)$-dimensional patch.

between similar structures. Fig. 3.3(c) and 3.3(d) show two likelihood maps f and f' with identical segmentations, both with incorrect topology comparing with the ground truth g (Fig. 3.3(b)). However, f is preferable as we need much less effort to change it so that the thresholded segmentation X has a correct topology. In particular, look closely to Fig. 3.3(c) and 3.3(d) near the broken handles and view the landscape of the function. To restore the broken handle in Fig. 3.3(d), we need to spend more effort to fill a much deeper gap than in Fig. 3.3(c). The same situation happens near the missing bridge between the two connected components.

To capture such subtle structural differences between different likelihood maps, we need a holistic view. In particular, we use the theory of *persistent homology* [19, 20]. Instead of choosing a fixed threshold, persistent homology theory captures all possible topological structures from all thresholds, and summarizes all these information in a concise format, called a *persistence diagram*.



(a) Segmentation (b) GT (c) Likelihood f (d) Likelihood f'

Figure 3.3: Illustration of topology and topology of a likelihood. For visualization purposes, the higher the function values are, the darker the area is. **(a)** an example segmentation X with two connected components and one handle. **(b)** The ground truth with one connected component and two handles. It can also be viewed as a binary valued function g. **(c)** a likelihood map f whose segmentation (bounded by the red curve) is X. The landscape views near the broken bridge and handle are drawn. Critical points are highlighted in the segmentation. **(d)** another likelihood map f' with the same segmentation as f. But the landscape views reveal that f' is worse than f due to deeper gaps.

Fig. 3.3 shows that only considering one threshold $\alpha = 0.5$ is insufficient. We consider thresholding the likelihood function with all possible thresholds. The thresholded results, $f^\alpha := \{x \in \Omega | f(x) \geq \alpha\}$ at different α's, constitute a filtration, i.e., a monotonically growing sequence induced by decreasing the threshold $\alpha : \varnothing \subseteq f^{\alpha_1} \subseteq f^{\alpha_2} \subseteq ... \subseteq f^{\alpha_n} = \Omega$, where $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_n$. As α decreases, the topology of f^α changes. Some new topological structures are

born while existing ones are killed. When $\alpha < \alpha_n$, only one connected component survives and never gets killed. See Fig. 3.4(a) and 3.4(d) for filtrations induced by the ground truth g (as a binary-valued function) and the likelihood f.

For a continuous-valued function f, its *persistence diagram*, $\mathrm{Dgm}(f)$, contains a finite number of dots in 2-dimensional plane, called *persistent dots*. Each persistent dot $p \in \mathrm{Dgm}(f)$ corresponds to a topological structure born and dies in the filtration. Denote by $\mathrm{birth}(p)$ and $\mathrm{death}(p)$ the birth and death time/threshold of the structure. For the connected component born at a global minimum and never dies, we say it dies at $\max_x f(x) = 1$. The coordinates of the dot p in the diagram are $(1 - \mathrm{birth}(p), 1 - \mathrm{death}(p))$. [3] Fig. 3.4(b) and 3.4(e) show the diagrams of g and f, respectively. Instead of comparing discrete Betti numbers, we can use the information from persistence diagrams to compare a likelihood f with the ground truth g in terms of topology.

To compute $\mathrm{Dgm}(f)$, we use the classic algorithm [19, 20] with an efficient implementation [162, 163]: we first discretize an image patch into vertices (pixels), edges, and squares. Note we adopt a cubical complex discretization, which is more suitable for images. The adjacency relationship between these discretized elements and their likelihood function values are encoded in a boundary matrix, whose rows and columns correspond to vertices/edges/squares. The matrix is reduced using a modified Gaussian elimination algorithm. The pivoting entries of the reduced matrix correspond to all the dots in $\mathrm{Dgm}(f)$. This algorithm is cubic to the matrix dimension, which is linear to the image size.

3.2.2 Topological Loss and its Gradient

We are now ready to formalize the topological loss, which measures the topological similarity between the likelihood f and the ground truth g. We abuse the notation and also view g as a binary valued function. We use the dots in the persistence diagram of f as they capture all possible topological structures f potentially has. We slightly modify the *Wasserstein distance* for persistence diagrams [112, 164, 165]. For persistence diagrams $\mathrm{Dgm}(f)$ and $\mathrm{Dgm}(g)$, we find the best one-to-one correspondence between the two sets of dots, and measure the total squared distance between them. [4] An unmatched dot will be matched to the diagonal line.

[3] Unlike traditional setting, we use $1 - \mathrm{birth}$ and $1 - \mathrm{death}$ as the x and y axes, because we are using an upperstar filtration, i.e., using the superlevel set, and decreasing α value.

[4] To be exact, the matching needs to be done on separate dimensions. Dots of 0-dim structures (blue markers in Fig. 3.4(b) and 3.4(e)) should be matched to the diagram of 0-dim structures. Dots of 1-dim structures (red markers in Fig. 3.4(b) and 3.4(e)) should be matched to the diagram of

Fig. 3.4(c) shows the optimal matching of the diagrams of g and f. Fig. 3.4(f) shows the optimal matching of $\mathrm{Dgm}(g)$ and $\mathrm{Dgm}(f')$. The latter is clearly more expensive.

The matching algorithm is as follows. A total of k (=Betti number) dots from ground truth ($\mathrm{Dgm}(g)$) are at the upper-left corner $p_{ul} = (0, 1)$, with $\mathrm{birth}(p_{ul}) = 1$ and $\mathrm{death}(p_{ul}) = 0$ (Fig. 3.4(b)). In $\mathrm{Dgm}(f)$, we find the k dots closest to the corner p_{ul} and match them to the ground truth dots. The remaining dots in $\mathrm{Dgm}(f)$ are matched to the diagonal line. The algorithm computes and sorts the squared distances from all dots in $\mathrm{Dgm}(f)$ to p_{ul}. The complexity is $O(n \log n)$, n = the number of dots in $\mathrm{Dgm}(f)$. In general, the state-of-the-art matches two arbitrary diagrams in $O(n^{3/2})$ time [166].

Let Γ be the set of all possible bijections between $\mathrm{Dgm}(f)$ and $\mathrm{Dgm}(g)$. The loss $L_{topo}(f, g)$ is:

$$\min_{\gamma \in \Gamma} \sum_{p \in \mathrm{Dgm}(f)} ||p - \gamma(p)||^2 = \sum_{p \in \mathrm{Dgm}(f)} [\mathrm{birth}(p) - \mathrm{birth}(\gamma^*(p))]^2 + [\mathrm{death}(p) - \mathrm{death}(\gamma^*(p))]^2 \tag{3.2}$$

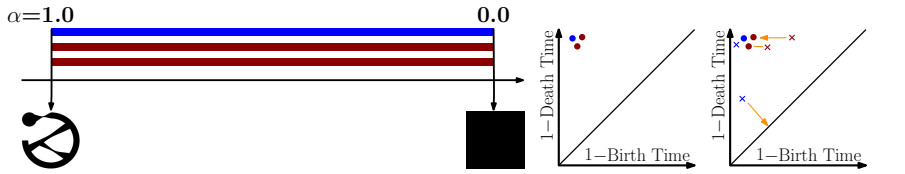
where γ^* is the optimal matching between two different point sets.

Intuitively, this loss measures the minimal amount of necessary effort to modify the diagram of $\mathrm{Dgm}(f)$ to $\mathrm{Dgm}(g)$ by moving all dots toward their matches. Note there are more dots in $\mathrm{Dgm}(f)$ (Fig. 3.4(c)) than in $\mathrm{Dgm}(g)$ (Fig. 3.4(b)); there will usually be some noise in the predicted likelihood map. If a dot p cannot be matched, we match it to its projection on the diagonal line, $\{(1 - b, 1 - d) | b = d\}$. This means we consider it as noise that should be removed. The dots matched to the diagonal line correspond to small noisy components or noisy loops. These dots will be pushed to the diagonal. And their corresponding components/loops will be removed or merged with others.

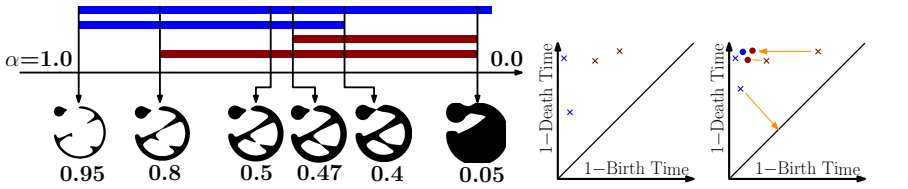
In this example, the extra connected component (a blue cross) in $\mathrm{Dgm}(f)$ will be removed. For comparison, we also show in Fig. 3.4(f) the matching between diagrams of the worse likelihood f' and g. The cost of the matching is obviously higher, i.e., $L_{topo}(f', g) > L_{topo}(f, g)$. As a theoretical reassurance, it has been proven that this metric for diagrams is stable, and the loss function $L_{topo}(f, g)$ is Lipschitz with regard to the likelihood function f [167].

The following theorem guarantees that the topological loss, when minimized to zero, enforces the constraint that the segmentation has the same topology and the ground truth.

1-dim structures.



(a) Filtration induced by the ground truth function, g. (b) $\mathrm{Dgm}(g)$ (c) $\mathrm{Dgm}(g)$+$\mathrm{Dgm}(f)$



(d) Filtration induced by the likelihood function, f. (e) $\mathrm{Dgm}(f)$ (f) $\mathrm{Dgm}(g)$+$\mathrm{Dgm}(f')$

Figure 3.4: An illustration of persistent homology. **Left** the filtrations on the ground truth function g and the likelihood function f. The bars of blue and burgundy colors are connected components and handles respectively. **(a)** For g, all structures are born at $\alpha = 1.0$ and die at $\alpha = 0$. **(d)** For f, from left to right, the birth of two components, birth of the longer handle, segmentation at $\alpha = 0.5$, the birth of the shorter handle, death of the extra component, death of both handles. **(b)** and **(e)** the persistence diagrams of g and f. **(c)** the overlay of the two diagrams. Orange arrows denote the matching between the persistent dots. The extra component (a blue cross) from the likelihood is matched to the diagonal line and will be removed if we move $\mathrm{Dgm}(f)$ to $\mathrm{Dgm}(g)$. **(f)** the overlay of the diagrams of g and the worse likelihood $\mathrm{Dgm}(f')$. The matching is obviously more expensive.

Theorem 1 (Topological Correctness)**.** *When the loss function $L_{topo}(f, g)$ is zero, the segmentation by thresholding f at 0.5 has the same Betti number as g.*

Proof. Assume $L_{topo}(f, g)$ is zero. By Eq. 3.2, $\mathrm{Dgm}(f)$ and $\mathrm{Dgm}(g)$ are matched perfectly, i.e., $p = \gamma^*(p), \forall p \in \mathrm{Dgm}(f)$. The two diagrams are identical and have the same number of dots.

Since g is a binary-valued function, as we decrease the threshold α continuously, all topological structures are created at $\alpha = 1$. The number of topological structures (Betti number) of g^α for any $0 < \alpha < 1$ is the same as the number of dots in $\mathrm{Dgm}(g)$. Note that for any $\alpha \in (0, 1)$, g^α is the ground truth segmentation. Therefore, the Betti number of the ground truth is the number of dots in $\mathrm{Dgm}(g)$. Similarly, for any $\alpha \in (0, 1)$, the Betti number of f^α equals to the number of dots

in $\mathrm{Dgm}(f)$. Since the two diagrams $\mathrm{Dgm}(f)$ and $\mathrm{Dgm}(g)$ are identical, the Betti number of the segmentation $f^{0.5}$ is the same as the ground truth segmentation. [5]

Topological Gradient. The loss function (Eq. 3.2) depends on crucial thresholds at which topological changes happen, e.g., birth and death times of different dots in the diagram. These crucial thresholds are uniquely determined by the locations at which the topological changes happen. When the underlying function f is differentiable, these crucial locations are exactly *critical points*, i.e., points with zero gradients. In the training context, our likelihood function f is a piecewise-linear function controlled by the neural network predictions at pixels. For such f, a critical point is always a pixel, since topological changes always happen at pixels. Denote by ω the neural network parameters. For each dot $p \in \mathrm{Dgm}(f)$, we denote by $c_b(p)$ and $c_d(p)$ the birth and death critical points of the corresponding topological structure (See Fig. 3.3(c) for examples).

Formally, we can show that the gradient of the topological loss $\nabla_\omega L_{topo}(f, g)$ is:

$$\begin{aligned}\sum_{p\in\mathrm{Dgm}(f)} 2[f(c_b(p)) - \mathrm{birth}(\gamma^*(p))]\frac{\partial f(c_b(p))}{\partial \omega} \\ +2[f(c_d(p)) - \mathrm{death}(\gamma^*(p))]\frac{\partial f(c_d(p))}{\partial \omega}\end{aligned} \tag{3.3}$$

To see this, within a sufficiently small neighborhood of f, any other piecewise linear function will have the same super level set filtration as f. The critical points of each persistent dot in $\mathrm{Dgm}(f)$ remain constant within such a small neighborhood. So does the optimal mapping γ^*. Therefore, the gradient can be straightforwardly computed based on the chain rule, as Eq. 3.3. When function values at different vertices are the same, or when the matching is ambiguous, the gradient does not exist. However, these cases constitute a measure of zero subspace in the space of likelihood functions. In summary, $L_{topo}(f, g)$ is a piecewise differentiable loss function over the space of all possible likelihood functions f.

Intuition. During training, we take the negative gradient, i.e., $-\nabla_\omega L_{topo}(f, g)$. For each topological structure, the gradient descent step is pushing the corresponding dot $p \in \mathrm{Dgm}(f)$ toward its match $\gamma^*(p) \in \mathrm{Dgm}(g)$. These coordinates are the function values of the critical points $c_b(p)$ and $c_d(p)$. They are both moved closer to the matched persistent dot in $\mathrm{Dgm}(g)$. We also show the negative gradient force in the landscape view of the function f (blue arrow in Fig. 3.3(c)). Intuitively, forcing from the topological gradient will push the saddle points up so that the broken bridge gets connected.

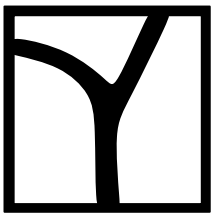
[5] Note that a more careful proof should be done for diagrams of 0- and 1-dimension separately.

3.2.3 Training a Neural Network

We present some crucial details of our training algorithm. Although our method is architecture-agnostic, we select one architecture inspired by DIVE [1], which was designed for neuron image segmentation tasks. Our network contains six trainable weight layers, four convolutional layers, and two fully connected layers. The first, second, and fourth convolutional layers are followed by a single max pooling layer of size 2×2 and stride 2 by the end of the layer. Particularly, because of the computational complexity, we use a patch size of 65×65 during the training process.

We use small patches (65×65) instead of big patches/whole images. The reason is twofold. First, the computation of topological information is relatively expensive. Second, the matching process between the persistence diagrams of predicted likelihood maps and ground truth can be quite difficult. For example, if the patch size is too big, there will be many persistent dots in $\mathrm{Dgm}(g)$ and even more dots in $\mathrm{Dgm}(g)$. The matching process is too complex and prone to errors. *By focusing on smaller patches, we localize topological structures and fix them one by one.*

Topology of Small Patches and Relative Homology. The small patches (65×65) often only contain partial branching structures rather than closed loops. To have a meaningful topological measure on these small patches, we apply *relative persistent homology* as a more localized approach for the computation of topological structures. Particularly, for each patch, we consider the topological structures relative to the boundary. It is equivalent to adding a black frame to the boundary and computing the topology to avoid trivial topological structures. As shown in the figure on the right, with the additional frame, a Y-shaped branching structure cropped within the patch will create two handles and be captured by persistent homology.



Training using this localized topological loss can be very efficient via random patch sampling. Specifically, we do not partition the image into patches. Instead, we randomly and densely sample patches that can overlap. As Theorem 1 guarantees, Our loss enforces correct topology within each sampled patch. These overlaps between patches propagate correct topology everywhere. On the other hand, correct topology within a patch means the segmentation can be a deformation of the ground truth. But the deformation is constrained within the patch. The patch size controls the tolerable geometric deformation. During training, even for the same patch, the

diagram $\mathrm{Dgm}(f)$, the critical pixels, and the gradients change. At each epoch, we resample patches and reevaluate their persistence diagrams, and the loss gradients. After computing topological gradients of all sampled patches from a mini-batch, we aggregate them for backpropagation.

3.3 Experiments

3.3.1 Datasets

We evaluate our method on six natural and biomedical datasets: **CREMI** [6], **ISBI12** [168], **ISBI13** [169], **CrackTree** [170], **Mass.** [171] and **DRIVE** [172].

- **CREMI** contains 125 images of size 1250x1250.
- **ISBI12** [168] contains 30 images of size 512x512.
- **ISBI13** [169] contains 100 images of size 1024x1024.
- **CrackTree** [170] contains 206 images of cracks in road (resolution 600x800).
- **Mass.** [171] has 1108 images from the Massachusetts Roads Dataset. The resolution is 1500x1500.
- **DRIVE** [172] is a retinal vessel segmentation dataset with 20 images. The resolution is 584x565.

The first three are neuron image segmentation datasets. The task is to segment membranes and eventually partition the image into neuron regions. For all datasets, we use a three-fold cross-validation and report the mean performance over the validation set.

3.3.2 Evaluation Metrics

We use four different evaluation metrics.

- **Pixel-wise accuracy** is the percentage of correctly classified pixels.

[6] https://cremi.org/

- **Betti number error** directly compares the topology (number of handles) between the segmentation and the ground truth. [7] We randomly sample patches over the segmentation and report the average absolute difference between their Betti numbers and the corresponding ground truth patches.
- **Adapted Rand Index (ARI)** is the maximal F-score of the foreground-restricted Rand index, a measure of similarity between two clusters. On this version of the Rand index, we exclude the zero component of the original labels (background pixels of the ground truth).
- **Variation of Information (VOI)** is a measure of the distance between two clusterings. It is closely related to mutual information; indeed, it is a simple linear expression involving mutual information.

The first one is pixel-wise evaluation metric, and the remaining three metrics are topology-relevant.

3.3.3 Baselines

We compare the proposed method with three popular baselines.

- **DIVE** [1] is a state-of-the-art neural network that predicts the probability of every individual pixel in a given image being a membrane (border) pixel or not.
- **UNet** [6] is a popular image segmentation method trained with cross-entropy loss.
- **UNet-VGG** [94] uses the response of selected filters from a pretrained CNN to construct the topology aware loss. For all methods, we generate segmentations by thresholding the predicted likelihood maps at 0.5.

3.3.4 Results

Tab. 3.1 shows the quantitative results for three different neuron image datasets, ISBI12, ISBI13, and CREMI. Tab. 3.2 shows the quantitative results for DRIVE, CrackTree, and Mass.. Our method significantly outperforms existing methods

[7] Note we focus on 1-dimensional topology in evaluation and training as they are more crucial in practice.

Table 3.1: Quantitative results for different models on several medical datasets.

Dataset	Method	Accuracy	ARI	VOI	Betti Error
ISBI12	**DIVE**	0.9640	0.9434	1.235	3.187
	UNet	**0.9678**	0.9338	1.367	2.785
	UNet-VGG	0.9532	0.9312	0.983	1.238
	TopoNet	0.9626	**0.9444**	**0.782**	**0.429**
ISBI13	**DIVE**	**0.9642**	0.6923	2.790	3.875
	UNet	0.9631	0.7031	2.583	3.463
	UNet-VGG	0.9578	0.7483	1.534	2.952
	TopoNet	0.9569	**0.8064**	**1.436**	**1.253**
CREMI	**DIVE**	**0.9498**	0.6532	2.513	4.378
	UNet	0.9468	0.6723	2.346	3.016
	UNet-VGG	0.9467	0.7853	1.623	1.973
	TopoNet	0.9456	**0.8083**	**1.462**	**1.113**

in topological accuracy (in all three topology-aware metrics), without sacrificing pixel accuracy. Fig. 3.5 shows qualitative results. Our method demonstrates more consistency in terms of structures and topology. It correctly segments fine structures such as membranes, roads, and vessels, while all other methods fail to do so. Note that the topological error cannot be solved by training with dilated ground truth masks. We run additional experiments on the CREMI dataset by training a topology-agnostic model with dilated ground truth masks. For 1 and 2 pixel dilation, We have Betti Error 4.126 and 4.431, respectively. They are still significantly worse than TopoLoss (Betti Error = 1.113).

3.3.5 Ablation Study: Loss Weights

Our loss (Eq. 3.1) is a weighted combination of cross entropy loss and topological loss. For convenience, we drop the weight of cross entropy loss and weight the topological loss with λ. Fig. 3.6(b) and 3.6(c) show ablation studies of λ on CREMI w.r.t. accuracy, Betti error and convergence rate. As we increase lambda, per-pixel accuracy is slightly compromised. The Betti error decreases at first but increases later. One important observation is that a certain amount of topological loss improves the convergence rate significantly. Empirically, we choose λ via cross-validation. Different datasets have different λ's. In general, λ is at the magnitude of 1/10000. This is understandable; while cross entropy loss gradient

Table 3.2: Quantitative results for different models on some other datasets.

Dataset	Method	Accuracy	ARI	VOI	Betti Error
DRIVE	**DIVE**	**0.9549**	0.8407	1.936	3.276
	UNet	0.9452	0.8343	1.975	3.643
	UNet-VGG	0.9543	0.8870	1.167	2.784
	TopoNet	0.9521	**0.9024**	**1.083**	**1.076**
CrackTree	**DIVE**	**0.9854**	0.8634	1.570	1.576
	UNet	0.9821	0.8749	1.625	1.785
	UNet-VGG	0.9833	0.8897	1.113	1.045
	TopoNet	0.9826	**0.9291**	**0.997**	**0.672**
Mass.	**DIVE**	0.9734	0.8201	2.368	3.598
	UNet	**0.9786**	0.8189	2.249	3.439
	UNet-VGG	0.9754	0.8456	1.457	2.781
	TopoNet	0.9728	**0.8671**	**1.234**	**1.275**

is applied to all pixels, the topological gradient is only applied to a sparse set of critical pixels. Therefore, the weight needs to be much smaller to avoid overfitting with these critical pixels.

Fig. 3.6(a) shows the weighted topological loss (λL_{topo}), cross entropy loss (L_{bce}) and total loss (L) at different training epochs. After 30 epochs, the total loss becomes stable. Meanwhile, while L_{bce} increases slightly, L_{topo} decreases. This is reasonable; incorporating of topological loss may force the network to overtrain on certain locations (near critical pixels), and thus may hurt the overall pixel accuracy slightly. This is confirmed by the pixel accuracy of TopoLoss in Tab. 3.1 and Tab. 3.2.

3.3.6 Rationale of the Proposed Algorithm

To further explain the rationale of topological loss, we first study an example training patch. In Fig. 3.7, we plot the likelihood map and the segmentation at different epochs. Within a short period, the likelihood map and the segmentation are stabilized globally, mostly thanks to the cross-entropy loss. After epoch 20, topological errors are gradually fixed by the topological loss. Notice the change of the likelihood map is only at specific topology-relevant locations.

Our topological loss compliments cross-entropy loss by combating sampling bias. In Fig. 3.7, for most membrane pixels, the network learns to make correct

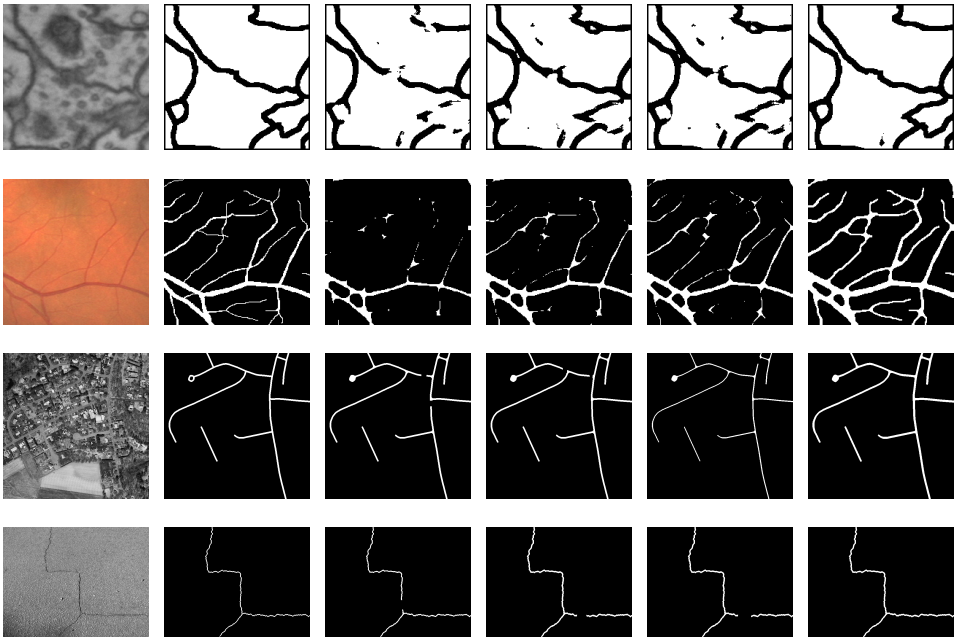
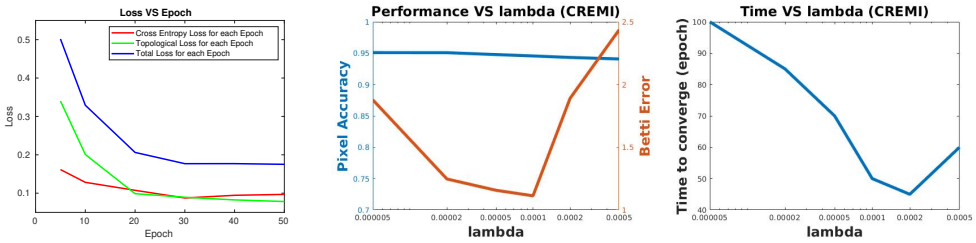


Figure 3.5: Qualitative results of the proposed method compared to other models. From left to right, sample images, ground truth, results for **DIVE**, **UNet**, **UNet-VGG** and our proposed **TopoLoss**.

predictions quickly. However, for a small number of difficult locations (blurred regions), it is much harder to learn to predict correctly. The issue is these locations only take a small portion of training pixel samples. Such disproportion cannot be changed even with more annotated training images. The topological loss essentially identifies these difficult locations during training (as critical pixels). It then forces the network to learn patterns near these locations, at the expense of overfitting and consequently slightly compromised per-pixel accuracy. On the other hand, we stress that topological loss cannot succeed alone. Without cross-entropy loss, inferring topology from a completely random likelihood map is meaningless. Cross-entropy loss finds a reasonable likelihood map so that the topological loss can improve its topology.



(a) Loss Curve (b) Performance VS λ (c) Time VS λ

Figure 3.6: **(a)** Cross Entropy loss, Topological loss, and total loss in terms of training epochs. **(b)** Ablation studies of lambda on CREMI w.r.t. accuracy, Betti error. **(c)** Ablation study of lambda on CREMI w.r.t. convergence rate.

3.4 Conclusion

In this chapter, we introduce a new topological loss driven by persistent homology and incorporate it into the end-to-end training of deep neural networks. Our method is particularly suitable for fine structure image segmentation. Quantitative and qualitative results show that the proposed topological loss term helps to achieve better performance in terms of topological-relevant metrics. Our proposed topological loss term is generic and can be incorporated into different deep learning architectures.

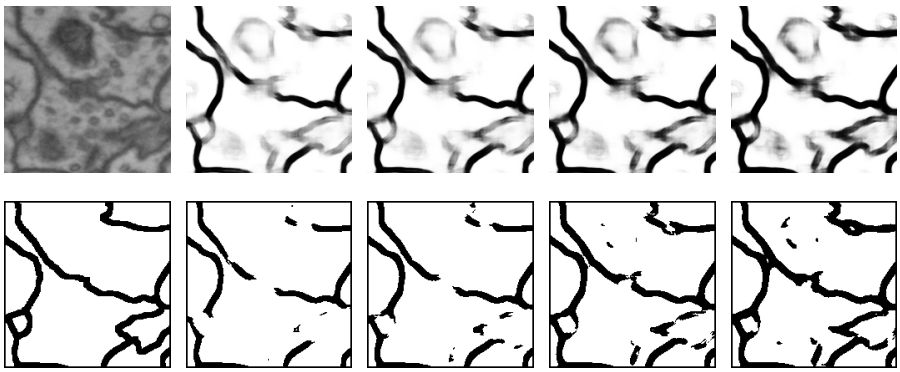


Figure 3.7: For a sample patch from CREMI, we show the likelihood map and segmentation at different training epochs. The first row corresponds to likelihood maps and the second row are thresholded results. From left to right, the original patch/ground truth, results after 10, 20, 30, and 40 epochs.

Chapter 4

Trigger Hunting with a Topological Prior for Trojan Detection

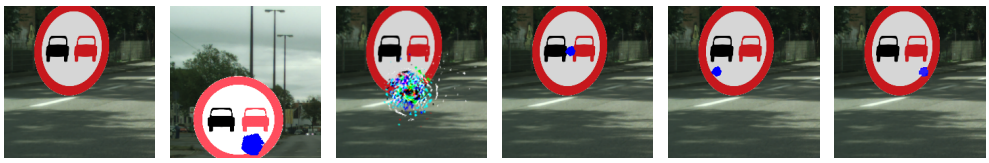
In the previous chapter, we introduced the proposed differentiable persistent-homology based topological loss and demonstrated its effectiveness in image segmentation tasks. In this chapter, we further explore the power of the topological loss in a quite different context, i.e., trojan detection.

4.1 Introduction

Deep learning has achieved superior performance in various computer vision tasks, such as image classification [79], image segmentation [10], object detection [173], etc. However, the vulnerability of DNNs against backdoor attacks raises serious concerns. In this chapter, we address the problem of *Trojan attacks*, where during training, an attacker injects *polluted samples*. While resembling normal samples, these polluted samples contain a specific type of perturbation (called triggers). These polluted samples are assigned with *target labels*, which are usually different from the expected class labels. Training with this polluted dataset results in a *Trojaned model*. At the inference stage, a Trojaned model behaves normally given clean samples. But when the trigger is present, it makes unexpected, yet consistently incorrect predictions.

One major constraint for Trojan detection is the limited access to polluted training data. In practice, the end-users, who need to detect the Trojaned models, often only have access to the weights and architectures of the trained DNNs. State-of-the-art (SOTA) Trojan detection methods generally adopt a *reverse engineering*

approach [3–5, 143, 145, 146]. They start with a few clean samples, using either gradient descent or careful stimuli crafting, to find a potential trigger that alters model prediction. Characteristics of the recovered triggers along with the associated network activations are used as features to determine whether a model is Trojaned or not.



(a) (b) (c) (d) (e) (f)

Figure 4.1: Illustration of recovered triggers: **(a)** clean image, **(b)** poisoned image, **(c)** image with a trigger recovered without topological prior, **(d)-(f)** images with candidate triggers recovered with the proposed method. Topological prior contributes to improved compactness. We run the trigger reconstruction for multiple rounds with a diversity prior to ensuring a diverse set of trigger candidates.

Trojan triggers can be of arbitrary patterns (e.g., shape, color, texture) at arbitrary locations of an input image (e.g., Fig. 4.1). As a result, one major challenge of the reverse engineering-based approach is the enormous search space for potential triggers. Meanwhile, just like the trigger is unknown, the target label (i.e., the class label to which a triggered model predicts) is also unknown in practice. Gradient descent may flip a model's prediction to the closest alternative label, which may not necessarily be the target label. This makes it even more challenging to recover the true trigger. Note that many existing methods [3, 5, 145] require a target label. These methods achieve target label independence by enumerating all possible labels, which can be computationally prohibitive especially when the label space is huge.

We propose a novel target-label-agnostic reverse engineering method. First, to improve the quality of the recovered triggers, we need a prior that can localize the triggers, but in a flexible manner. We, therefore, propose to enforce a *topological prior* to the optimization process of reverse engineering, i.e., the recovered trigger should have fewer connected components. This prior is implemented through a topological loss based on the theory of persistent homology [19]. It allows the recovered trigger to have arbitrary shape and size. Meanwhile, it ensures the trigger is not scattered and is reasonably localized. See Fig. 4.1 for an example – comparing (d)-(f) vs. (c).

As a second contribution, we propose to reverse engineer *multiple diverse trigger candidates*. Instead of running gradient descent once, we run it for multiple rounds, each time producing one trigger candidate, e.g., Fig. 4.1 (d)-(f). Furthermore, we propose a *trigger diversity loss* to ensure the trigger candidates are sufficiently different from each other (see Fig. 4.2). Generating multiple diverse trigger candidates can increase the chance of finding the true trigger. It also mitigates the risk of unknown target labels. In the example of Fig. 4.2, the first trigger candidate flips the model prediction to a label different from the target, while only the third candidate hits the true target label.

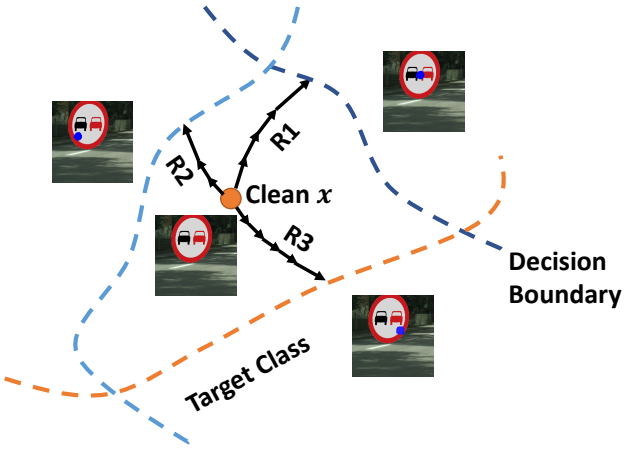


Figure 4.2: Illustration of generating a diverse set of trigger candidates to increase the chance of finding the true trigger, especially for scenarios with unknown target labels.

Generating multiple trigger candidates, however, adds difficulties in filtering out already-subtle cues for Trojan detection. We also note reverse engineering approaches often suffer from false positive triggers such as adversarial perturbations or direct modification of the crucial objects of the image. [1] In practice, we systematically extract a rich set of features to describe the characteristics of the reconstructed trigger candidates based on geometry, color, and topology, as well as network activations. A Trojan-detection network is then trained to detect Trojaned models based on these features. Our main contributions are summarized as follows:

- We propose a topological prior to regularize the optimization process of

[1] Modification of crucial objects is usually not a valid trigger strategy; it is too obvious to end-users and is against the principle of adversaries.

reverse engineering. The prior ensures the locality of the recovered triggers, while being sufficiently flexible regarding the appearance. It significantly improves the quality of the reconstructed triggers.

- We propose a diversity loss to generate multiple diverse trigger candidates. This increases the chance of recovering the true trigger, especially for cases with unknown target labels.
- Combining the topological prior and diversity loss, we propose a novel Trojan detection framework. On both synthetic and public TrojAI benchmarks, our method demonstrates substantial improvement in both trigger recovery and Trojan detection.

4.2 Method

Our reverse engineering framework is illustrated in Fig. 4.3. Given a trained DNN model, either clean or Trojaned, and a few clean images, we use gradient descent to reconstruct triggers that can flip the model's prediction. To increase the quality of reconstructed triggers, we introduce novel diversity loss and topological prior. They help recover multiple diverse triggers of high quality.

The common hypothesis of reverse engineering approaches is that the reconstructed triggers will appear different for Trojaned and clean models. To fully exploit the discriminative power of the reconstructed triggers for Trojan detection, we extract features based on trigger characteristics and associated network activations. These features are used to train a classifier, called the Trojan-detection network, to classify a given model as Trojaned or clean.

We note the discriminative power of the extracted trigger features, and thus the Trojan-detection network is highly dependent on the quality of the reconstructed triggers. Empirical results will show the proposed diversity loss and topological prior are crucial in reconstructing high quality triggers, and ensuring a high quality Trojan-detection network. We will show that our method can learn to detect Trojaned models even when trained with a small amount of labeled DNN models.

For the rest of this section, we mainly focus on the reverse engineering module. We also add details of the Trigger feature extraction to Sec. 4.2.3 and details of the Trojan-detection network to Sec. 4.2.4.

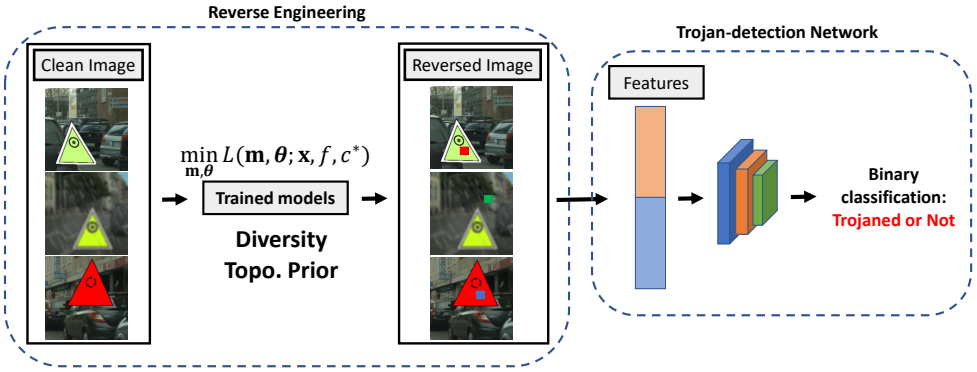


Figure 4.3: Our Trojan detection framework.

4.2.1 Reverse Engineering of Multiple Diverse Trigger Candidates

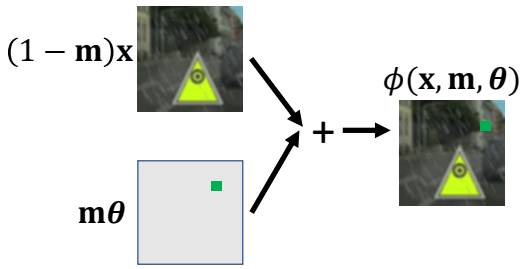


Figure 4.4: $\mathbf{m}$ and $\boldsymbol{\theta}$ convert an input image $\mathbf{x}$ into an altered one $\phi(\mathbf{x},\mathbf{m},\boldsymbol{\theta})$. The $\odot$ is omitted here for simplification.

Our method is based on the existing reverse engineering pipeline first proposed by Neural Cleanse [3]. Given a trained DNN model, let $f(\cdot)$ be the mapping from an input clean image $\mathbf{x} \in \mathbb{R}^{3\times M\times N}$ to the output $\mathbf{y} \in \mathbb{R}^K$ with K classes, where M and N denote the height and width of the image, respectively. Denote by $f_k(\cdot)$ the k-th output of f. The predicted label c^* is given by $c^* = \arg\max_k f_k(\mathbf{x})$, $1 \le k \le K$. We introduce parameters $\boldsymbol{\theta}$ and $\mathbf{m}$ to convert $\mathbf{x}$ into an altered sample

$$\phi(\mathbf{x},\mathbf{m},\boldsymbol{\theta}) = (\mathbf{1}-\mathbf{m}) \odot \mathbf{x} + \mathbf{m} \odot \boldsymbol{\theta}, \tag{4.1}$$

where the binary mask $\mathbf{m} \in \{0,1\}^{M\times N}$ and the pattern $\boldsymbol{\theta} \in \mathbb{R}^{M\times N}$ determine the trigger. $\mathbf{1}$ denotes an all-one matrix. The symbol "$\odot$" denotes Hadamard

product. See Fig. 4.4 for an illustration. We intend to find a triggered image $\hat{\mathbf{x}} = \phi(\mathbf{x}, \hat{\mathbf{m}}, \hat{\boldsymbol{\theta}})$ so that the model prediction $\hat{c} = \arg\max_k f_k(\hat{\mathbf{x}})$ is different from the prediction on the original image c^*.

We find the triggered image, $\hat{\mathbf{x}}$, by minimizing a loss over the space of $\mathbf{m}$ and $\boldsymbol{\theta}$:

$$L(\mathbf{m}, \boldsymbol{\theta}; \mathbf{x}, f, c^*) = L_{flip}(\ldots) + \lambda_1 L_{div}(\ldots) + \lambda_2 L_{topo}(\ldots) + R(\mathbf{m}), \quad (4.2)$$

where L_{flip}, L_{div}, and L_{topo} denote the label-flipping loss, diversity loss, and topological loss, respectively. We temporarily dropped their arguments for convenience. λ_1, λ_2 are the weights to balance the loss terms. $R(\mathbf{m})$ is a regularization term penalizing the size and range of the mask (more details will be provided in Sec. 4.2.4).

To facilitate optimization, we relax the constraint on the mask $\mathbf{m}$ and allow it to be a continuous-valued function, ranging between 0 and 1 and defined over the image domain, $\mathbf{m} \in [0, 1]^{M \times N}$. Next, we introduce the three loss terms one-by-one.

Label-Flipping Loss L_{flip}: The label-flipping loss L_{flip} penalizes the prediction of the model regarding the ground truth label, formally:

$$L_{flip}(\mathbf{m}, \boldsymbol{\theta}; \mathbf{x}, f, c^*) = f_{c^*}(\phi(\mathbf{x}, \mathbf{m}, \boldsymbol{\theta})). \quad (4.3)$$

Minimizing L_{flip} means minimizing the probability that the altered image $\phi(\mathbf{x}, \mathbf{m}, \boldsymbol{\theta})$ is predicted as c^*. In other words, we are pushing the input image out of its initial decision region.

Note that we do not specify which label we would like to flip the prediction to. This makes the optimization easier. Existing approaches often run optimization to flip the label to a target label and enumerate through all possible target labels [3, 145]. This can be rather expensive in computation, especially with large label space.

The downside of not specifying a target label during optimization is we will potentially miss the correct target label, i.e., the label that the Trojaned model predicts on a triggered image. To this end, we propose to reconstruct multiple candidate triggers with diversity constraints. This will increase the chance of hitting the correct target label. See Fig. 4.2 for an illustration.

Diversity Loss L_{div}: With the label-flipping loss L_{flip}, we flip the label to a different one from the original clean label and recover the corresponding triggers. The new label, however, may not be the same as the true target label. Also considering the huge trigger search space, it is difficult to recover the triggers with

only one attempt. Instead, we propose to search for multiple trigger candidates to increase the chance of capturing the true trigger.

We run our algorithm for N_T rounds, each time reconstructing a different trigger candidate. To avoid finding similar trigger candidates, we introduce the diversity loss L_{div} to encourage different trigger patterns and locations. Let $\mathbf{m}_j$ and $\boldsymbol{\theta}_j$ denote the trigger mask and pattern found in the j-th round. At the i-th round, we compare the current candidates with triggers from all previous founds in terms of L_2 norm. Formally:

$$L_{div}(\mathbf{m}, \boldsymbol{\theta}) = -\sum_{j=1}^{i-1} ||\mathbf{m} \odot \boldsymbol{\theta} - \mathbf{m}_j \odot \boldsymbol{\theta}_j||_2. \tag{4.4}$$

Minimizing L_{div} ensures the eventual trigger $\mathbf{m}_i \odot \boldsymbol{\theta}_i$ to be different from triggers from previous rounds. Fig. 4.1(d)-(f) demonstrates the multiple candidates recovered with sufficient diversity.

4.2.2 Topological Prior

Quality control of the trigger reconstruction remains a major challenge in reverse engineering methods, due to the huge search space of triggers. Even with the regularizer $R(\mathbf{m})$, the recovered triggers can still be scattered and unrealistic. See Fig. 4.1(c) for an illustration. We propose a topological prior to improve the locality of the reconstructed trigger. We introduce a topological loss enforcing that the recovered trigger mask $\mathbf{m}$ has as few connected components as possible. The loss is based on the theory of persistent homology [19, 20], which models the topological structures of a continuous signal in a robust manner. Note that the assumption holds if the trigger is not spanned over the whole image, which fits the scenarios we are discussing in our work.

Persistent Homology. We introduce persistent homology in the context of 2D images. A more comprehensive treatment of the topic can be found in [19, 174]. Recall we relaxed the mask function $\mathbf{m}$ to a continuous-valued function defined over the image domain (denoted by Ω). Given any threshold α, we can threshold the image domain with regard to $\mathbf{m}$ and obtain the *superlevel set*, $\Omega^\alpha := \{p \in \Omega | \mathbf{m}(p) \geq \alpha\}$. A superlevel set can have different topological structures, e.g., connected components and holes. If we continuously decrease the value α, we have a continuously growing superlevel set Ω^α. This sequence of superlevel set is called a *filtration*. The topology of Ω^α continuously changes through the filtration. New connected components are born and later die (merged with others). New holes are born and later die (sealed up). For each topological structure, the threshold at

which it is born is called its *birth time*. The threshold at which it dies is called its *death time*. The difference between birth and death time is called the *persistence* of the topological structure.

We record the lifespan of all topological structures over the filtration and encode them via a 2D point set called *persistence diagram*, denoted by $\mathrm{Dgm}(\mathbf{m})$. Each topological structure is represented by a 2D point within the diagram, $p \in \mathrm{Dgm}(\mathbf{m})$, called a *persistent dot*. We use the birth and death times of the topological structure to define the coordinates of the corresponding persistent dot. For each dot $p \in \mathrm{Dgm}(\mathbf{m})$, we abuse the notation and call the birth/death time of its corresponding topological structure as $\mathrm{birth}(p)$ and $\mathrm{death}(p)$. Then we have $p = (\mathrm{death}(p), \mathrm{birth}(p))$. See Fig. 4.5 for an example function $\mathbf{m}$ (viewed as a terrain function) and its corresponding diagram. There are five dots in the diagram, corresponding to five peaks in the landscape view.

To compute the persistence diagram, we use the classic algorithm [19, 20] with an efficient implementation [162, 163]. The image is first discretized into a cubical complex consisting of vertices (pixels), edges, and squares. A boundary matrix is then created to encode the adjacency relationship between these elements. The algorithm essentially carries out a matrix reduction algorithm over the boundary matrix, and the reduced matrix reads out the persistence diagram.

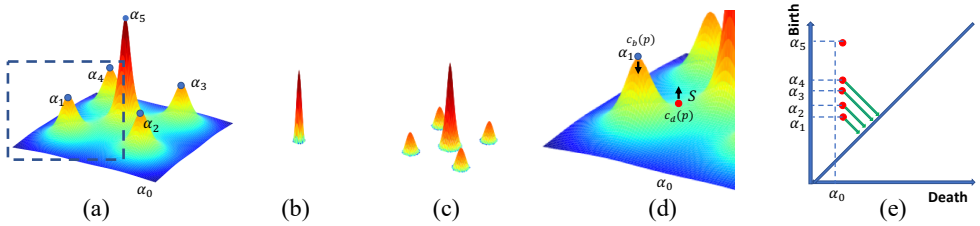


Figure 4.5: From the left to right: **(a)** a sample landscape for a continuous function. The values at the peaks $\alpha_0 < \alpha_1 < \alpha_2 < \alpha_3 < \alpha_4 < \alpha_5$. As we decrease the threshold, the topological structures of the superlevel set change, (**b**) and (**c**) correspond to topological structures captured by different thresholds, (**d**) highlighted region in (**a**), (**e**) the changes are captured by the persistence diagram (right figure). We focus on the 0-dimensional topological structures (connected components). Each persistent dot in the persistence diagram denotes a specific connected component. The topological loss is introduced to reduce the connected components, which means pushing most of the persistent dots to the diagonal (along the green lines).

Topological Loss L_{topo}: We formulate our topological loss based on the persistent

homology described above. Minimizing our loss reduces the number of connected components of triggers. We will focus on zero-dimensional topological structures, i.e., connected components. Intuitively speaking, each dot in the diagram corresponds to a connected component. The ones far away from the diagonal line are considered salient as their birth and death times are far apart. And the ones close to the diagonal line are considered trivial. In Fig. 4.5, there is one salient dot far away from the diagonal line. It corresponds to the highest peak. The other four dots are closer to the diagonal line and correspond to the smaller peaks. The topological loss will reduce the number of connected components by penalizing the distance of all dots from the diagonal line, except for the most salient one. Formally, the loss L_{topo} is defined as:

$$L_{topo}(\mathbf{m}) = \sum\nolimits_{p \in \mathrm{Dgm}(m) \setminus \{p^*\}} [\mathrm{birth}(p) - \mathrm{death}(p)]^2, \tag{4.5}$$

where p^* denotes the persistent dot that is farthest away from the diagonal (with the highest persistence). Minimizing this loss will keep p^* intact, while pushing all other dots to the diagonal line, thus making their corresponding components either disappear or merged with the main component.

Differentiability and the Gradient: The loss function (Eq. 4.5) is differentiable almost everywhere in the space of functions. To see this, we revisit the filtration, i.e., the growing superlevel set as we continuously decrease the threshold α. The topological structures change at specific locations of the image domain. A component is born at the corresponding local maximum. It dies merging with another component at the saddle point between the two peaks. In fact, these locations correspond to critical points of the function. And the function values at these critical points correspond to the birth and death times of these topological structures. For a persistent dot, p, we call the critical point corresponding to its birth, $c_b(p)$, and the critical point corresponding to its death, $c_d(p)$. Then we have $\mathrm{birth}(p) = \mathbf{m}(c_b(p))$ and $\mathrm{death}(p) = \mathbf{m}(c_d(p))$. The loss function (Eq. 4.5) can be rewritten as a polynomial function of the function $\mathbf{m}$ at different critical points.

$$L_{topo}(\mathbf{m}) = \sum\nolimits_{p \in \mathrm{Dgm}(m) \setminus \{p^*\}} [\mathbf{m}(c_b(p)) - \mathbf{m}(c_d(p))]^2. \tag{4.6}$$

The gradient can be computed naturally. L_{topo} is a piecewise differentiable loss function over the space of all possible functions $\mathbf{m}$. In a gradient decent step, for all dots except for p^*, we push up the function at the death critical point $c_d(p)$ (the saddle), and push down the function value at the birth critical point $c_b(p)$ (the local maximum). This is illustrated by the arrows in Fig. 4.5 (Middle-Right). This will kill the non-salient components and push them toward the diagonal.

4.2.3 Trigger Feature Extraction and Trojan Detection Network

Next, we summarize the features we extract from recovered triggers. The recovered Trojan triggers can be characterized via their capability in flipping model predictions (i.e., the label-flipping loss). Moreover, they are different from adversarial noise as they tend to be more regularly shaped and are also distinct from actual objects which can be recognized by a trained model. We introduce appearance-based features to differentiate triggers from adversarial noise and actual objects.

Specifically, for label flipping capability, we directly use the label-flipping loss L_{flip} and diversity loss L_{div} as features. For appearance features, we use trigger size and topological statistics as their features: 1) The number of foreground pixels divided by the total number of pixels in mask $\mathbf{m}$; 2) To capture the size of the triggers in the horizontal and vertical directions, we fit a Gaussian distribution to the mask $\mathbf{m}$ and record *mean* and *std* in both directions; 3) The trigger we find may have multiple connected components. The final formulated topological descriptor includes the topological loss L_{topo}, the number of connected components, *mean*, and *std* in terms of the size of each component.

After the features are extracted, we build a neural network for Trojan detection, which takes the bag of features of the generated triggers as inputs, and outputs a scalar score of whether the model is Trojaned or not. More details are provided in Sec. 4.2.4.

4.2.4 Learning-based Trojan-detection Network

While bottom-up trigger generation uses appearance heuristics to search for possible triggers, we cannot guarantee the recovered triggers are true triggers, even for Trojaned models. Many other perturbations can create label flipping effects, such as adversarial samples and modifications specific to the most semantically critical region. See Fig. 4.6 for illustrations. These examples can easily become false positives for a Trojan detector; they can be good trigger candidates generated by the reverse engineer pipeline.

To fully address these issues, we propose a top-down Trojan detector learned using clean and Trojaned models. It helps separate true Trojan triggers from other false positives. To this end, we propose to extract features from the reverse engineered Trojan triggers and train a separate shallow neural network for Trojan detection.

For each model, we generate a diverse set of N_T possible triggers for each of its K output classes to scan for possible triggers. As described above, we extract one

feature for each generated trigger. As a result, for each model, we have $N_T \times K$ sets of features.

Regularizer: The regularizer $R(\mathbf{m})$ consists of a mass term and a size term. For mass, we use $\bar{\mathbf{m}}$ as the average value of $\mathbf{m}$. For size, we normalize the mask $\mathbf{m}$ into a distribution $p(x, y)$ over x and y. To capture the spatial extent of mask m, we compute the standard deviation of $X \sim p(x)$ as δ_X and $Y \sim p(y)$ as δ_Y. As a result, $R(\mathbf{m}) = \bar{\mathbf{m}} + \delta_X + \delta_Y$ is the regularizer.

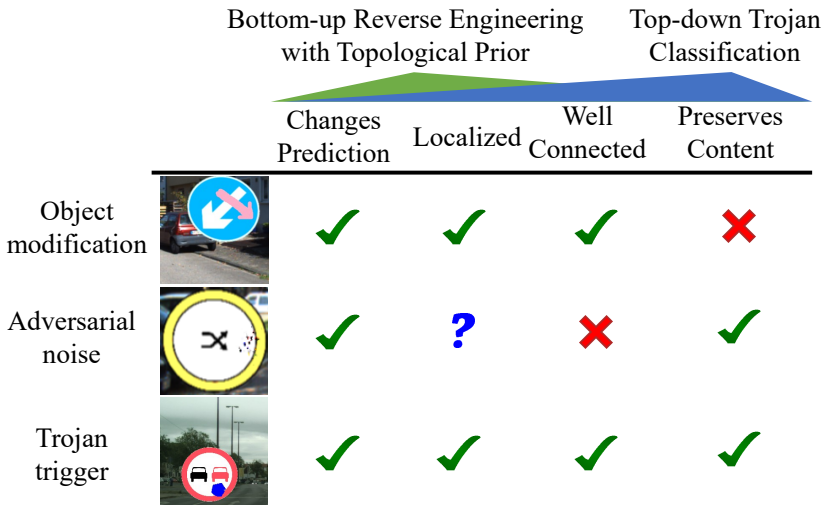


Figure 4.6: Our Trojan detection method combines bottom-up Trigger reverse engineering under topological constraints, with top-down classification. Such a combination allows us to accurately isolate Trojan triggers from non-Trojan patterns such as adversarial noise and object modifications.

Classification Network: After the features are extracted, we build a neural network for Trojan detection, which takes as input for a given image classifier, the bag of features of its generated triggers and outputs a scalar score of whether the model is Trojaned or not.

Since different models may have different numbers of output classes K, the number of features varies for each model. Therefore, a sequence of modeling architectures – such as bag-of-words, Recurrent Neural Networks, and Transformers – could be employed to aggregate the $N_T \times K$ features into a $0/1$ Trojan classification output.

Given a set of annotated models, clean or infected, supervised learning can be applied to train the classification network. Empirically, we found that a simple

bag-of-words technique achieved the best Trojan detection performance while also being fast to run and data efficient. Specifically, let the bag of features be $\{v_i\}$, $i = 1, \ldots, N_T K$. The features are first transformed individually using an MLP, followed by average pooling across the features and another MLP to output the Trojan classification:

$$\vec{h}_i = MLP_\alpha(\vec{v}_i), \quad \vec{h} = \frac{1}{N_T K} \sum_i \vec{h}_i, \quad s = MLP_\beta(\vec{h}), \quad i = 1, \ldots N_T K. \tag{4.7}$$

4.2.5 Supporting Additional Trigger Classes

Different classes of Trojan triggers are being actively explored in Trojan detection benchmarks. For image classification, the TrojAI datasets include localized triggers which can be directly applied to objects along with global filter-based Trojans, where the idea is that a color filter could be attached to the lens of the camera and results in a global image transformation.

Our top-down bottom-up Trojan detection framework is designed to support multiple classes of Trojan triggers, where each trigger class gets its dedicated reverse engineering approach and pathway in the Trojan detection network. Adding support to a new class of triggers, e.g. color filters, amounts to adding a reverse engineering approach for color filters and adding an appearance feature descriptor for Trojan classification.

Reverse Engineering Color Filter Triggers. The pipeline of reverse engineering color filter triggers is illustrated in Fig. 4.7. Compared to reverse engineering local triggers in Fig. 4.4, the filter editor and the loss functions are adjusted to find color filter triggers.

We model a color filter trigger using a per-pixel position-dependent color transformation. Let $[r_{ij}, g_{ij}, b_{ij}]$ be the color of a pixel of input image $\mathbf{x}$ at location (i, j), and we model a color filter trigger using an MLP $E(\cdot; \theta^{\text{filter}})$ with parameters θ^{filter} which performs position-dependent color mapping:

$$[\hat{r_{ij}}, \hat{g_{ij}}, \hat{b_{ij}}] = E([r_{ij}, g_{ij}, b_{ij}, i, j]; \theta^{\text{filter}}). \tag{4.8}$$

Here $[\hat{r_{ij}}, \hat{g_{ij}}, \hat{b_{ij}}]$ is the color of pixel (i, j) of the triggered image $\hat{\mathbf{x}}$. For TrojAI datasets we use a 2-layer 16 hidden neurons MLP to model the color filters, as it learns sufficiently complex color transforms while being fast to run:

$$L^{\text{filter}} = L^{\text{filter}}_{flip} + \lambda_1^{\text{filter}} L^{\text{filter}}_{div} + \lambda_2^{\text{filter}} R^{\text{filter}}(\theta^{\text{filter}}). \tag{4.9}$$

The label flipping loss L_{flip}^{filter} remains identical:

$$L_{flip}^{\text{filter}} = \hat{y}_c. \tag{4.10}$$

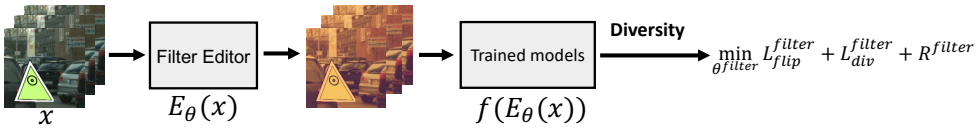


Figure 4.7: Reverse engineering of global color filter triggers.

The diversity loss L_{div}^{filter} is designed to induce diverse color transforms. We use how a color filter transforms 16 random (r, g, b, i, j) tuples to characterize a color filter $E(\cdot; \theta^{\text{filter}})$. We record the 16 output $(\hat{r}, \hat{g}, \hat{b})$ tuples as a 48-dim descriptor of the color filter, denoted as $u_{\theta^{\text{filter}}}$. The diversity loss is the L_2 norm of the current candidate to previously found triggers:

$$L_{div}^{\text{filter}} = -\sum_{j=1}^{N_T} \sum_{i=1}^{j-1} ||u_{\theta_i^{\text{filter}}} - u_{\theta_j^{\text{filter}}}||_2. \tag{4.11}$$

The regularizer term $R^{\text{filter}}(\theta^{\text{filter}})$ is simply an L2 regularizer on the MLP parameters to reduce the complexity on the color transforms:

$$R^{\text{filter}}(\theta^{\text{filter}}) = ||\theta^{\text{filter}}||_2^2. \tag{4.12}$$

Feature Extractor for Color Filter Triggers. For each model we use reverse engineering to generate K classes by N_T^{filter} diverse color filter triggers. For each color filter trigger, we combine an appearance descriptor with label flipping loss L_{flip}^{filter} and diversity loss L_{div}^{filter} as its combined feature descriptor. For the appearance descriptor, we use the same descriptor discussed in diversity: how a color filter $E(\cdot; \theta^{\text{filter}})$ transforms 16 random (r, g, b, i, j) tuples. We record the 16 output $(\hat{r}, \hat{g}, \hat{b})$ tuples as a 48-dim appearance descriptor.

As a result for each model, we have $N_T^{filter} K$ features for color filter triggers.

Trojan Classifier with Color Filter Triggers. A bag-of-words model is used to aggregate those features. The aggregated features across multiple trigger classes, e.g. color filters and local triggers, are concatenated and fed through an MLP for final Trojan classification as below:

$$\vec{h}_i^{filter} = MLP_\alpha^{filter}(\vec{v}_i^{filter}), \quad \vec{h}^{filter} = \frac{1}{N_T^{filter} K} \sum_i \vec{h}_i^{filter}, \quad i = 1, \dot{N}_T^{filter} K \tag{4.13}$$

$$\vec{h}_i^{local} = MLP_\alpha^{local}(\vec{v}_i^{local}), \quad \vec{h}^{local} = \frac{1}{N_T K} \sum_i \vec{h}_i^{local}, \quad i = 1, \dots N_T K \tag{4.14}$$

$$s = MLP_\beta([\vec{h}^{filter}; \vec{h}^{local}]), \tag{4.15}$$

For the TrojAI datasets, color filter reverse engineering is conducted using Adam optimizer with a learning rate 3×10^{-2} for 10 iterations. Hyperparameters are set to $\lambda_1^{\text{filter}} = 0.05$ and $\lambda_2^{\text{filter}} = 10^{-4}$. We also set $N_T = 2$ and $N_T^{filter} = 8$.

4.3 Experiments

We evaluate our method on both synthetic datasets and publicly available TrojAI benchmarks. We provide quantitative and qualitative results, followed by ablation studies, to demonstrate the efficacy of the proposed method. All clean/Trojaned models are DNNs trained for image classification.

4.3.1 Datasets

Synthetic Datasets (Trojaned-MNIST and Trojaned-CIFAR10): We adopt the codes provided by NIST [2] to generate 200 DNNs (50% of them are Trojaned) trained to classify MNIST and CIFAR10 data, respectively. The Trojaned models are trained with images poisoned by square triggers. The poison rate is set as 0.2. **TrojAI Benchmarks (TrojAI-Round1, Round2, Round3 and Round4)**: These datasets are provided by US IARPA/NIST [3], who recently organized a Trojan AI competition. Polygon triggers are generated randomly with variations in shape, size, and color. Filter-based triggers are generated by randomly choosing from five distinct filters. Trojan detection is more challenging on these TrojAI datasets as compared to Triggered-MNIST due to the use of deeper DNNs and larger variations in the appearances of foreground/background objects, trigger patterns, etc. Round1, Round2, Round3, and Round4 have 1000, 1104, 1008, and 1008 models, respectively.

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

[3] https://pages.nist.gov/trojai/docs/data.html

These datasets contain trained models for traffic sign classification (for each round, 50% of total models are Trojaned). All the models are trained on synthetically created image data of non-real traffic signs superimposed on road background scenes. Trojan detection is more difficult on Round2/Round3/ Round4 compared to Round1 due to following reasons:

- Round2/Round3 have more number of classes: Round1 has 5 classes while Round2/Round3 have 5-25 classes.
- Round2/Round3 have more trigger types: Round1 only has polygon triggers, while Round2/Round3 have both polygon and Instagram filter based triggers.
- The number of source classes are different: all classes are poisoned in Round1, while 1, 2, or all classes are poisoned in Round2/Round3.
- Round2/Round3 have more type of model architectures: Round1 has 3 architectures, while Round2/Round3 have 23 architectures.

Round3 experimental design is identical to Round2 with the addition of Adversarial Training. Two different Adversarial Training approaches: Projected Gradient Descent (PGD), and Fast is Better than Free (FBF) [175] are used.

Unlike the previous rounds, Round4 can have multiple concurrent triggers. Additionally, triggers can have conditions attached to their firing. The differences are listed as follows:

- All triggers in Round4 are one to one mappings, which means a trigger flips a single source class to a single target class.
- Three possible conditionals, spatial, spectral, and class are attached to triggers within this dataset.
- Round4 has removed the very large model architectures to reduce the training time.

Round1, Round2, Round3, and Round4 have 1000, 1104, 1008, and 1008 models respectively.

4.3.2 Evaluation Metrics

We follow the settings in [176]. We report the mean and standard deviation of two metrics: area under the ROC curve (AUC) and accuracy (ACC). Specifically, we

evaluate our approach on the whole set by doing an 8-fold cross validation. For each fold, we use 80% of the models for training, 10% for validation, and the rest 10% for testing.

4.3.3 Baselines

We carefully choose the methods with available codes from the authors as our baselines, including NC (Neural Cleanse) [3], ABS [4], TABOR [5], ULP [144], and DLTND [145]. We follow the instructions to obtain the reported results and list all the available repositories here:

- **Neural Cleanse**: https://github.com/bolunwang/backdoor
- **ABS**: https://github.com/naiyeleo/ABS
- **TABOR**: https://github.com/UsmannK/TABOR
- **ULP**: https://github.com/UMBCvision/Universal-Litmus-Patterns
- **DLTND**: https://github.com/wangren09/TrojanNetDetector

4.3.4 Implementation Details

We set $\lambda_1 = 1$, $\lambda_2 = 10$, and $N_T = 3$ for all our experiments (i.e., we generate 3 trigger candidates for each input image and each model). The parameters of the Trojan detection network are learned using a set of clean and Trojaned models with ground truth labeling. We train the detection network by optimizing cross entropy loss using the Adam optimizer [177]. The hidden state size, the number of layers of MLP_α, MLP_β, as well as optimizer learning rate, weight decay, and the number of epochs are optimized using Bayesian hyperparameter search [4] for 500 rounds on 8-fold cross-validation.

4.3.5 Results

Tab. 4.1 and Tab. 4.2 show the quantitative results on the Trojaned-MNIST/ CIFAR10 and TrojAI datasets, respectively. The reported performances of baselines are reproduced using source codes provided by the authors or quoted from related papers. The best performing numbers are highlighted in bold. From Tab. 4.1

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

and Tab. 4.2, we observe that our method performs substantially better than the baselines. It is also worth noting that, compared with these baselines, our proposed method extracts fix-sized features for each model, independent of the number of classes, architectures, trigger types, etc. By using the extracted features, we are able to train a separate Trojan detection network, which is salable and model-agnostic.

Table 4.1: Comparison on Trojaned-MNIST/CIFAR10.

Method	Metric	Trojaned-MNIST	Trojaned-CIFAR10
NC	AUC	0.57 ± 0.07	0.75 ± 0.07
ABS	AUC	0.63 ± 0.04	0.67 ± 0.06
TABOR	AUC	0.65 ± 0.07	0.71 ± 0.05
ULP	AUC	0.59 ± 0.03	0.55 ± 0.03
DLTND	AUC	0.62 ± 0.05	0.52 ± 0.08
Ours	AUC	$\mathbf{0.88 \pm 0.04}$	$\mathbf{0.91 \pm 0.05}$
NC	ACC	0.60 ± 0.04	0.73 ± 0.06
ABS	ACC	0.65 ± 0.02	0.69 ± 0.04
TABOR	ACC	0.62 ± 0.04	0.69 ± 0.08
ULP	ACC	0.57 ± 0.02	0.59 ± 0.06
DLTND	ACC	0.64 ± 0.07	0.55 ± 0.07
Ours	ACC	$\mathbf{0.89 \pm 0.02}$	$\mathbf{0.92 \pm 0.04}$

Fig. 4.8 shows a few examples of recovered triggers. We observe that, compared with the baselines, the triggers found by our method are more compact and of better quality. This is mainly due to the introduction of topological constraints. The improved quality of recovered triggers directly results in improved performance of Trojan detection.

4.3.6 Ablation Studies

Ablation Study of Loss Weights: For the loss weights λ_1 and λ_2, we empirically choose the weights which make reverse engineering converge the fastest. This is a reasonable choice as in practice, time is one major concern for reverse engineering pipelines.

Despite the seemingly ad hoc choice, we have observed that our performances are quite robust to all these loss weights. As topological loss is a major contribution of this chapter, we conduct an ablation study in terms of its weight (λ_2) on TrojAI-Round4 dataset. The results are reported in Fig. 4.9. We observe that the

Table 4.2: Performance comparison on the TrojAI dataset.

Method	Metric	TrojAI-Round1	TrojAI-Round2	TrojAI-Round3	TrojAI-Round4
NC	AUC	0.50 ± 0.03	0.63 ± 0.04	0.61 ± 0.06	0.58 ± 0.05
ABS	AUC	0.68 ± 0.05	0.61 ± 0.06	0.57 ± 0.04	0.53 ± 0.06
TABOR	AUC	0.71 ± 0.04	0.66 ± 0.07	0.50 ± 0.07	0.52 ± 0.04
ULP	AUC	0.55 ± 0.06	0.48 ± 0.02	0.53 ± 0.06	0.54 ± 0.02
DLTND	AUC	0.61 ± 0.07	0.58 ± 0.04	0.62 ± 0.07	0.56 ± 0.05
Ours	AUC	$\mathbf{0.90 \pm 0.02}$	$\mathbf{0.87 \pm 0.05}$	$\mathbf{0.89 \pm 0.04}$	$\mathbf{0.92 \pm 0.06}$
NC	ACC	0.53 ± 0.04	0.49 ± 0.02	0.59 ± 0.07	0.60 ± 0.04
ABS	ACC	0.70 ± 0.04	0.59 ± 0.05	0.56 ± 0.03	0.51 ± 0.05
TABOR	ACC	0.70 ± 0.03	0.68 ± 0.08	0.51 ± 0.05	0.55 ± 0.06
ULP	ACC	0.58 ± 0.07	0.51 ± 0.03	0.56 ± 0.04	0.57 ± 0.04
DLTND	ACC	0.59 ± 0.04	0.61 ± 0.05	0.65 ± 0.04	0.59 ± 0.06
Ours	ACC	$\mathbf{0.91 \pm 0.03}$	$\mathbf{0.89 \pm 0.04}$	$\mathbf{0.90 \pm 0.03}$	$\mathbf{0.91 \pm 0.04}$

proposed method is quite robust to λ_2, and when $\lambda = 10$, it achieves slightly better performance (AUC: 0.92 ± 0.06) than other choices.

Ablation Study of Number of Training Model Samples: The trigger features and Trojan detection network are important in achieving SOTA performance. To further demonstrate the efficacy of the proposed diversity and topological loss terms, we conduct another ablation study to investigate the case with fewer training model samples, and thus a weaker Trojan detection network.

The ablation study in terms of the number of training samples on TrojAI-Round4 data is illustrated in Tab. 4.3. We observe that the proposed topological loss and diversity loss will boost the performance with/without a fully trained Trojan-detection network. These two losses improve the quality of the recovered trigger, in spite of how the trigger information is used. Thus even with a limited number of training samples (e.g., 25), the proposed method could still achieve significantly better performance than the baselines.

Ablation Study for Loss Terms: We investigate the individual contribution of different loss terms used to search for latent triggers. Tab. 4.4 lists the corresponding performance on the TrojAI-Round4 dataset. We observe a decrease in AUC (from 0.92 to 0.89) if the topological loss is removed. This drop is expected as the topological loss helps to find more compact triggers. Also, the performance drops significantly (from 0.92 to 0.85 in AUC) if the diversity loss is removed. We also report the performance by setting $N_T = 2$; when $N_T = 2$, the performance increases from 0.85 to 0.89 in AUC. The reason is that with diversity loss, we are able to generate multiple diverse trigger candidates, which increases the probability

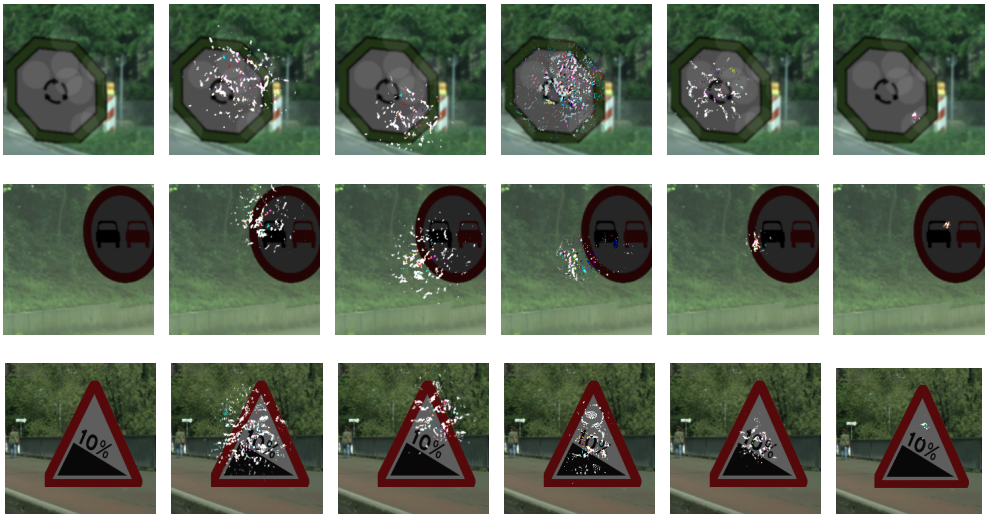


Figure 4.8: Examples of recovered triggers overlaid on clean images. From left to right: **(a)** clean image, **(b)** triggers recovered by [3], **(c)** triggers recovered by [4], **(d)** triggers recovered by [5], **(e)** triggers recovered by our method without topological prior, and **(f)** triggers recovered by our method with topological prior.

of recovering the true trigger when the target class is unknown. Our ablation study justifies the use of both diversity and topological losses.

In practice, we found that topological loss can improve the convergence of trigger searches. Without topological loss, it takes ≈50 iterations to find a reasonable trigger (Fig. 4.8(e)). In contrast, with the topological loss, it takes only ≈30 iterations to converge to a better recovered trigger (Fig. 4.8(f)). The rationale is that, as the topological loss imposes strong constraints on the number of connected components, it largely reduces the search space of triggers, consequently, making the convergence of trigger search much faster. This is worth further investigation.

4.3.7 Unsupervised Setting for Trojan Detection

Indeed, our method outperforms existing methods in different settings: fully supervised settings with annotated models, and unsupervised settings. In (Tab. 4.3), we have already demonstrated that with a limited number of annotated models, our method outperformed others. To make a fair comparison, following Neural Cleanse and DLTND, we also use the simple technique based on Median Absolute

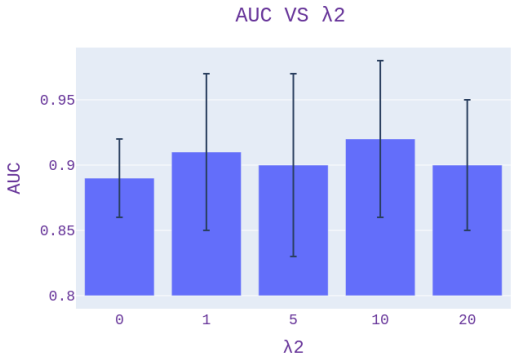


Figure 4.9: Ablation study results for λ_2.

Table 4.3: Ablation study for # of training samples.

# of samples	Ours	w/o topo	w/o diversity
25	**0.77 ± 0.04**	0.73 ± 0.03	0.68 ± 0.04
50	**0.81 ± 0.03**	0.76 ± 0.05	0.73 ± 0.02
100	**0.84 ± 0.05**	0.78 ± 0.06	0.76 ± 0.03
200	**0.86 ± 0.04**	0.82 ± 0.04	0.79 ± 0.05
400	**0.90 ± 0.05**	0.85 ± 0.03	0.82 ± 0.04
800	**0.92 ± 0.06**	0.89 ± 0.04	0.85 ± 0.02

Deviation (MAD) for trojan detection. We report the performance of Round 4 data of TrojAI in Tab. 4.5.

Because of the better trigger quality, due to the proposed losses, our method outperforms baselines such as Neural Cleanse. We also note that, in Tab. 4.5, all methods (including ours) perform unsatisfactorily in the unsupervised setting. This brings us back to the discussion as to whether a supervised setting is justified in Trojan detection (although this is not directly relevant to our method).

From the research point of view, we believe that data-driven methods for Trojan detection are unavoidable as the attack techniques continue to develop. Like in many other security research problems, the Trojan attack and defense are two sides of the same problem that are supposed to advance together. When the problem was first studied, classic unsupervised methods such as Neural Cleanse are sufficient. In recent years, the attack techniques have continued to develop, exploiting the entire dataset and leveraging techniques like adversarial training. Meanwhile, detection

Table 4.4: Ablation results of loss terms.

Method	TrojAI-Round4
w/o topological loss	0.89 ± 0.04
w/o diversity loss ($N_T = 1$)	0.85 ± 0.02
N_T = 2	0.89 ± 0.05
with all loss terms ($N_T = 3$)	$\mathbf{0.92 \pm 0.06}$

Table 4.5: Unsupervised performances on Trojan.

Method	**AUC**	**ACC**
NC	0.58	0.60
ABS	0.53	0.51
TABOR	0.52	0.55
ULP	0.54	0.57
DLTND	0.56	0.59
Ours	**0.63**	**0.65**

methods are confined to only the given model and a few sample data. For defense methods to move forward and to catch up with the attack methods, it seems only natural and necessary to exploit supervised approaches, e.g., learning patterns from public datasets such as the TrojAI benchmarks.

4.4 Conclusion

In this chapter, we propose a diversity loss and a topological prior to improve the quality of the trigger reverse engineering for Trojan detection. These loss terms help to find high quality triggers efficiently. They also avoid the dependence of the method on the target label. On both synthetic datasets and publicly available TrojAI benchmarks, our approach recovers high quality triggers and achieves SOTA Trojan detection performance.

Chapter 5

Structure-Aware Image Segmentation with Homotopy Warping

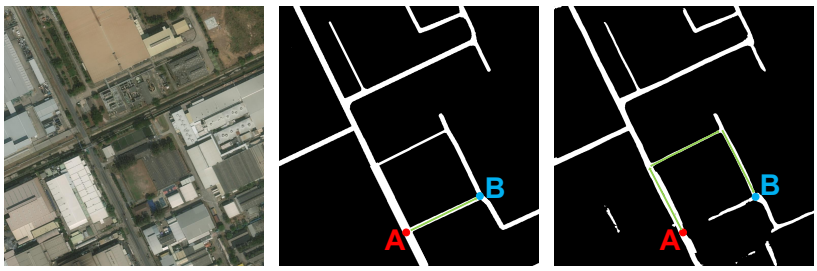
In Chapter 3, we introduced the persistent-homology based topological loss. The results are promising, while the identified critical points are very noisy and are often not relevant to the topological errors. In this chapter, we introduce another warping strategy to efficiently identify critical points.

5.1 Introduction

Image segmentation with topological correctness is a challenging problem, especially for images with fine-scale structures, e.g., satellite images, neuron images and vessel images. Deep learning methods have delivered strong performances in image segmentation tasks [10–14]. However, even with satisfying per-pixel accuracy, most existing methods are still prone to topological errors, i.e., broken connections, holes in 2D membranes, missing connected components, etc. These errors may significantly impact downstream tasks. For example, the reconstructed road maps from satellite images can be used for navigation [178–187]. A small amount of pixel errors will result in broken connections, causing incorrect navigation routes. See Fig. 5.1 for an illustration. In neuron reconstruction [55, 123–125, 188], the incorrect topology of the neuron membrane will result in erroneous merge or split of neurons, and thus errors in morphology and connectivity analysis of neuronal circuits.

Topological errors usually happen at challenging locations, e.g., weak connections or blurred locations. But not all challenging locations are topologically relevant; for example, pixels at the boundary of the object of interest can generally be challenging, but not relevant to topology. To truly suppress topological errors, we need to focus on *topologically critical pixels*, i.e., challenging locations that are topologically relevant. Without identifying and targeting these locations, neural networks that are optimized for standard pixel-wise losses (e.g., cross-entropy loss or mean-square-error loss) cannot avoid topological errors, even if we increase the training set size.

Existing works have targeted these topologically critical pixels. The closest method to our work is TopoNet [2], which is based on persistent homology [19,20]. The main idea is to identify topologically critical pixels corresponding to critical points of the likelihood map predicted by the neural network. The selected critical points are reweighed in the training loss to force the neural network to focus on them, and thus to avoid topological errors. But there are two main issues with this approach: 1) The method is based on the likelihood map, which can be noisy with a large amount of irrelevant critical points. This leads to inefficient optimization during training. 2) The computation for persistent homology is cubic to the image size. It is too expensive to recompute at every iteration.



(a) Image (b) GT mask (c) UNet pred.

Figure 5.1: An illustration for the importance of topological correctness. If one wants to go to point B from A, the shortest path in the GT is illustrated in **(b)** (the green path). However, in the result predicted by UNet, though only a few pixels are misclassified, the shortest path from A to B is totally different, which is illustrated by the green path in **(c)**. Zoom-in for better viewing.

In this chapter, we propose a novel approach to identify topologically critical pixels in a more efficient and accurate manner. These locations are penalized in the proposed *homotopy warping loss* to achieve better topological accuracy. Our

method is partially inspired by the warping error previously proposed to evaluate the topological accuracy [189]. Given a binary predicted mask f_B and a ground truth mask g, we "warp" one towards another without changing its topology. From the view of topology, to warp mask f_B towards g, we find a mask f_B^* that is homotopy equivalent to f_B and is as close to g as possible [190]. The difference between the warped mask f_B^* and g constitutes the topologically critical pixels. We can also warp g towards f_B and find another set of topologically critical pixels. See Fig. 5.4 and Fig. 5.5 for illustrations. These locations directly correspond to the topological difference between the prediction and the ground truth, tolerating geometric deformations. Our homotopy warping loss targets them to fix the topological errors of the model.

The warping of a mask is achieved by iteratively flipping labels at pixels without changing the topology of the mask. These flippable pixels are called *simple points/pixels* in the classic theory of digital topology [191]. Note that in this chapter, we focus on the topology of binary masks, simple points, and simple pixels can be used interchangeably. Finding the optimal warping of a mask towards another mask is challenging due to the huge search space. To this end, we propose a new heuristic method that is computationally efficient. We filter the image domain with the distance transform and flip simple pixels based on their distance from the mask. This algorithm is proven efficient and delivers high quality locally optimal warping results.

Overall, our contributions can be summarized as follows:

- We propose a novel *homotopy warping loss*, which penalizes errors on topologically critical pixels. These locations are defined by homotopic warping of predicted and ground truth masks. The loss can be incorporated into the training of topology-preserving deep segmentation networks.
- By exploiting distance transforms of binary masks, we propose a novel homotopic warping algorithm to identify topologically critical pixels in an efficient manner. This is essential in incorporating the homotopy warping loss into the training of deep nets.

Our loss is a plug-and-play loss function. It can be used to train any segmentation network to achieve better performance in terms of topology. We conduct experiments on both 2D and 3D benchmarks to demonstrate the efficacy of the proposed method. Our method performs strongly in multiple topology-relevant metrics (e.g., ARI, Warping Error, and Betti Error). We also conduct several ablation studies to further demonstrate the efficiency and effectiveness of the technical contributions.

5.2 Method

By warping the binary predicted mask to the ground truth or conversely, we can accurately and efficiently identify the critical pixels. And then we propose a novel homotopy warping loss that targets them to fix topological errors of the model. The overall framework is illustrated in Fig. 5.2.

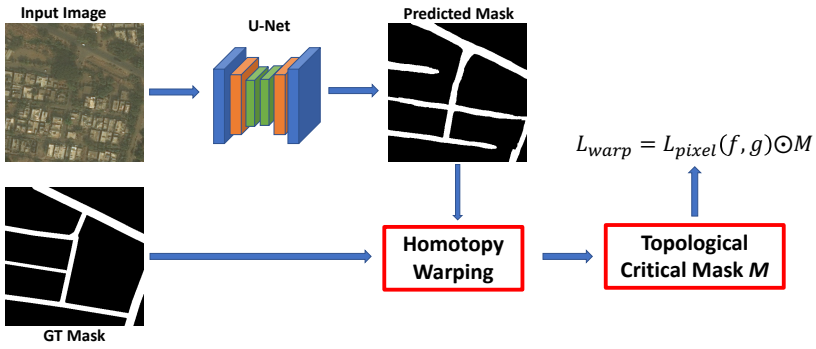


Figure 5.2: The illustration of the proposed *homotopy warping loss* L_{warp}. The homotopy warping algorithm tries to identify the topological critical pixels via the binary mask instead of noisy likelihood maps. These identified topological critical pixels/masks are used to define a new loss that is complementary to standard pixel-wise loss functions. The details of *Homotopy Warping* and *Topological Critical Mask M* can be found in Sec. 5.2.3 and Sec. 5.2.4, respectively.

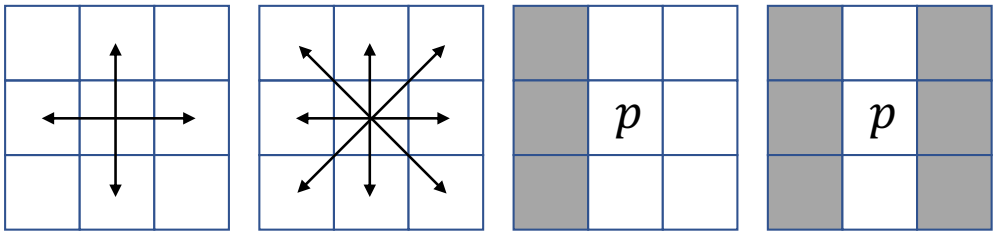
This section is organized as follows. We will start with the necessary definitions and notations. In Sec. 5.2.1, we give a concise description of digital topology and simple points. Next, we analyze different types of warping errors in Sec. 5.2.3. The proposed warping loss is introduced in Sec. 5.2.4. And finally, we explain the proposed new warping algorithm in Sec. 5.2.5.

5.2.1 Digital Topology and Simple Points

In this section, we briefly introduce the simple point definition from the classic digital topology [191]. We focus on the 2D setting, whereas all definitions generalize to 3D. Details on 3D images are provided in Sec. 5.2.2.

Connectivities of Pixels. To discuss the topology of a 2D binary image, we first define the connectivity between pixels. See Fig. 5.3 for an illustration. A pixel p has 8 pixels surrounding it. We can either consider the 4 pixels that share an

edge with p as p's neighbors (called *4-adjacency*), or consider all 8 pixels as p's neighbors (called *8-adjacency*). To ensure the Jordan closed curve theorem holds, one has to use one adjacency for foreground (FG) pixels, and the other adjacency for the background (BG) pixels. In this chapter, we use 4-adjacency for FG and 8-adjacency for BG. For 3D binary images, we use 6-adjacency for FG and 26-adjacency for BG. Denote by $N_4(p)$ the set of 4-adjacency neighbors of p, and $N_8(p)$ the set of 8-adjacency neighbors of p.



(a) 4-adjacent (b) 8-adjacent (c) Simple Point (d) Non-simple

Figure 5.3: Illustration for 4, 8-adjacency, simple and non-simple points. **(a)**: 4-adjacency. **(b)**: 8-adjacency. **(c)**: a simple point p. White and grey pixels are FG and BG, respectively. Flipping the label of p will not change the topology. **(d)**: a non-simple point p. Flipping p will change the topology.

Simple Points. For a binary image (2D/3D), a pixel/voxel is called a *simple point* if it could be flipped from foreground (FG) to background (BG), or from BG to FG, without changing the topology of the image [191]. The following definition can be used to determine whether a point is simple:

Definition 1 (Simple Point Condition [191]) *Let p be a point in a 2D binary image. Denote by F the set of FG pixels. Assume 4-adjacency for FG and 8-adjacent for BG. p is a simple point if and only if both of the following conditions hold: 1) p is 4-adjacent to just one FG connected component in $N_8(p)$, and 2) p is 8-adjacent to just one BG connected component in $N_8(p)$.*

See Fig. 5.3 for an illustration of simple and non-simple points in 2D cases. It is easy to check if a pixel p is simple or not by inspecting its 3×3 neighboring patch. The **Definition 1** can also generalize to the 3D setting with 6- and 26-adjacencies for FG and BG, respectively.

5.2.2 Simple Points in 3D

For a 3D binary image, a voxel is called a *simple point* if it could be flipped from foreground (FG) to background (BG), or from BG to FG, without changing the topology of the image [191, 192]. The following definition is a natural extension of the **Definition 1** in 2D case. It can be used to determine whether a voxel is simple:

Definition 2 (Simple Point Condition [191]) *Let p be a point in a 3D binary image. Denote by F the set of FG pixels. Assume 6-adjacency for FG and 26-adjacent for BG.*

5.2.3 Homotopic Warping Error

In this section, we introduce the homotopic warping of one mask towards another. We warp a mask through a sequence of flipping of simple points. Since we only flip simple points, by definition the warped mask will have the same topology. [1] The operation is called a *homotopic warping*. It has been proven that two binary images with the same topology can always be warped into each other by flipping a sequence of simple points [193].

Consider two input masks, the prediction mask, and the ground truth mask. We can warp one of them (source mask) into another (target mask) in the best way possible, i.e., the warped mask has a minimal number of differences from the target mask (formally, the minimal Hamming distance). Once the warping is finished, the pixels at which the warped mask is different from the target mask, called *critical pixels*, are a sparse set of pixels indicative of the topological errors of the prediction mask.

We will warp in both directions: from the prediction mask to the ground truth mask, and the opposite. They identify different sets of critical pixels for the same topological error. In Fig. 5.4, we show a synthetic example with red and white masks, as well as warping in both directions. Warping the red mask towards the white mask ((a) and (b)) results in a single-pixel wide gap. The pixels in the gap (highlighted with red crosses) are critical pixels; flipping any of them will change the topology of the warped red mask. Warping the white mask towards the red mask ((d) and (e)) results in a single pixel wide link connecting the warped white mask. All pixels along the link (highlighted with red crosses) are critical; flipping any of them will change the topology of the warped white mask.

[1] Note that it is essential to flip these simple points sequentially. The simple/non-simple status of a pixel may change if other adjacent pixels are flipped. Therefore, flipping a set of simple points *simultaneously* is not necessarily topology-preserving.

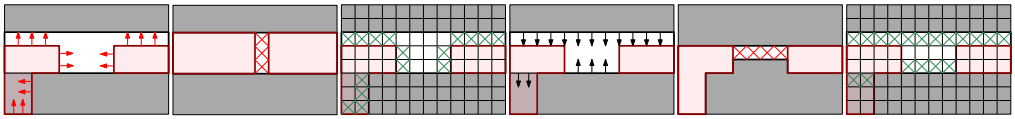
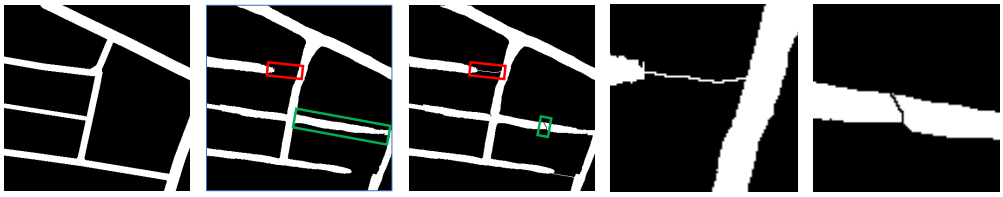


Figure 5.4: Illustration of homotopic warping between two masks, red and white. If red is the FG of the prediction, this is a false negative topological error; if red is the FG of the ground truth, this is a false positive error. **(a-c):** warping the red mask towards the white. **(a):** arrows show the warping direction. **(b):** the final mask after warping. Only a single-pixel wide gap remains in the middle of the warped red mask. The non-simple/critical pixels are highlighted with red crosses. They correspond to the topological error and will be penalized for the loss. **(c):** At the beginning of the warping, we highlight (with green crosses) simple points that can be flipped according to our algorithm. **(d-f):** warping the white mask towards the red mask. Only a single-pixel wide connection remains to ensure the warped white mask is connected. The non-simple/critical pixels are highlighted with red crosses.

Here if the red mask is the prediction mask, then this corresponds to a false negative connection, i.e., a connection that is missed by the prediction. If the red mask is the ground truth mask, then this corresponds to a false positive connection. Note the warping ensures that *only topological errors are represented by the critical pixels*. In the synthetic example (Fig. 5.4), the large area of error in the bottom left corner of the image is completely ignored as it is not topologically relevant.

In Fig. 5.5, we show a real example from the satellite image dataset, focusing on the errors related to 1D topological structures (connection). In the figure, we illustrate both a false negative connection error (highlighted with a red box) and a false positive connection error (highlighted with a green box). If we warp the ground truth mask towards the prediction mask (c), we observe critical pixels forming a link for the false negative connection (d), and a gap for the false positive connection (e). Similarly, we can warp the prediction mask towards the ground truth, and get different sets of critical pixels for the same topological errors. See Fig. 5.6.

Note that for 2D images with fine structures, errors regarding 1D topological structures are the most crucial. They affect the connectivity of the prediction results. For 3D images, errors on 1D or 2D topological structures are both important, corresponding to broken connections for tubular structures and holes in membranes.



(a) GT (b) Pred. (c) Warp GT (d) Zoom-in (e) Zoom-in

Figure 5.5: Illustration of warping in a real world example (satellite image). **(a)** GT mask. **(b)** The prediction mask. The red box highlights a *false negative connection*, and the green box highlights a *false positive connection*. **(c)** Warped GT mask (using the prediction mask as the target). **(d)** Zoomed-in view of the red box in **(c)**. **(e)** Zoomed-in view of the green box in **(c)**.

We also provide a more comprehensive characterization of different types of topological structures and errors in Fig. 5.7.

Note that for 3D vessel data, it's also the 1D topological structures (connections) that matter. The process of identifying the topological critical pixels is the same as Fig. 5.4 and Fig. 5.5. And the only difference is to use 6-adjacency for FG and 26-adjacency for BG.

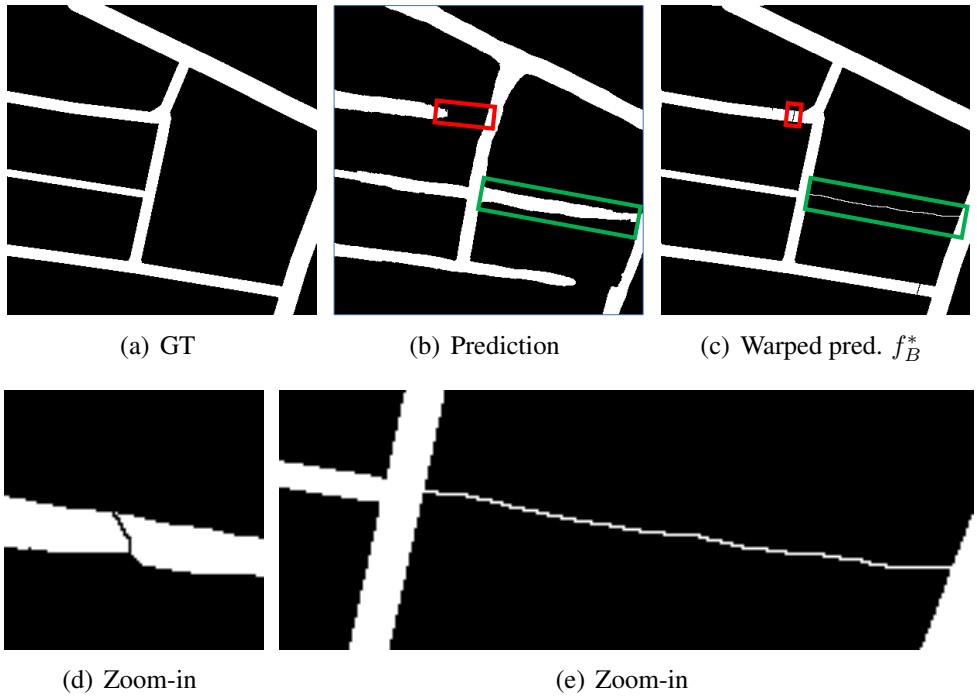
5.2.4 Homotopy Warping Loss

Next, we formalize the proposed *homotopy warping loss*, which is evaluated on the critical pixels due to homotopic warping. As illustrated in the previous section, the warping can be in both directions, from the prediction mask to the ground truth mask, and the opposite.

Formally, we denote by f the predicted likelihood map of a segmentation network, and f_B the corresponding binarized prediction mask (i.e., f thresholded at 0.5). We denote by g the ground truth mask. First, we warp g towards f_B, so that the warped mask, g^* has the minimal Hamming distance from the target f_B.

$$g^* = \arg\min_{g^w \lhd g} ||f_B - g^w||_H \tag{5.1}$$

where $\lhd$ is the homotopic warping operation. The pixels at which g^* and f_B disagree are the critical pixels and will be penalized in the loss. We record these critical pixels due to the warping of g with a mask M_g, $M_g = f_B \oplus g^*$, in which $\oplus$ is the *Exclusive Or* operation.



(a) GT (b) Prediction (c) Warped pred. f_B^*

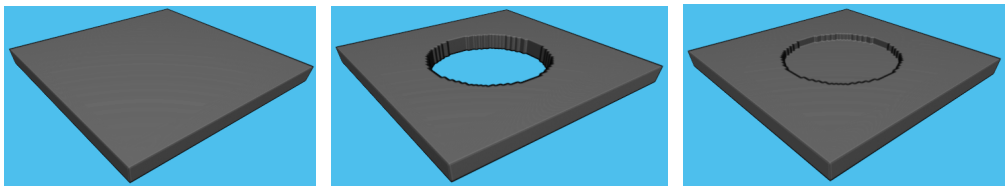
(d) Zoom-in (e) Zoom-in

Figure 5.6: Illustration of warping in a real world example (satellite image). **(a)** GT mask. **(b)** The prediction mask. The red box highlights a *false negative connection*, and the green box highlights a *false positive connection*. **(c)** Warped prediction mask (using the GT mask as the target). **(d)** Zoomed-in view of the red box in **(c)**. **(e)** Zoomed-in view of the green box in **(c)**.

We also warp the prediction mask f_B towards g.

$$f_B^* = \arg\min_{f_B^w \lhd f_B} ||f_B^w - g||_H \tag{5.2}$$

We use the mask M_f to record the remaining critical pixels after warping f_B, $M_f = g \oplus f_B^*$. The union of the two critical pixel masks is the complete set of critical pixels corresponding to topological errors, $M = M_g \cup M_f$. M contains all the locations directly related to topological structures. Note this is different from persistent-homology-based method [2], DMT based method [8] or skeleton based method [109], which extract topological locations/structures on the predicted continuous likelihood maps. Our warping loss directly locates the topological critical pixels/structures/locations based on the binary mask. The detected critical pixel set is sparse and less noisy.



(a) GT boundary (b) Pred. binary boundary (c) Warped GT boundary

Figure 5.7: Illustration of 2D topological structures (holes/voids) for 3D case. **(a)**: GT boundary. **(b)**: Prediction binary boundary. **(c)**: Warped GT boundary. If we warp the GT boundary (Fig.**(a)**) towards the prediction binary boundary (Fig.**(b)**), there will be a plane with a thickness of 1 in the middle of the hole/void to keep the original structure, which is illustrated in Fig. **(c)**.

L_{pixel} denotes the pixel-wise loss function (e.g., cross-entropy), then L_{warp} can be defined as:

$$L_{warp} = L_{pixel}(f, g) \odot M \tag{5.3}$$

where $\odot$ denotes Hadamard product. L_{warp} penalizes the topological critical pixels, forcing the neural network to predict better at these locations, and thus are less prone to topological errors.

The final loss of our method, L_{total}, is given by:

$$L_{total} = L_{dice} + \lambda_{warp} L_{warp} \tag{5.4}$$

where L_{dice} denotes the dice loss. And the loss weight λ_{warp} is used to balance the two loss terms.

5.2.5 Distance-Ordered Homotopy Warping

Even though checking whether a pixel is simple or not is easy, finding the optimal warping as in Eq. 5.1 and Eq. 5.2 is challenging. The reason is that there are too many degrees of freedom. At each iteration during the warping, we have to choose a simple point to flip. It is not obvious which simple point will finally lead to a global optimum.

In this section, we provide an efficient heuristic algorithm to find a warping local optimum. We explain the algorithm for warping g towards f_B. The algorithm generalizes to the opposite warping direction naturally. Recall the warping algorithm iteratively flips simple points. But there are too many choices at each

iteration. It is hard to know which flipping choice will lead to the optimal solution. We need good heuristics for choosing a flippable pixel. Below we explain our main intuitions for designing our algorithm.

First, we restrict the warping so it only sweeps through the area where the two masks disagree. In other words, at each iteration, we restrict the candidate pixels for flipping to not only simple but also pixels on which g and f_B disagree. In Fig. 5.4 (c) and (f), we highlight the candidate pixels for flipping at the beginning. Notice that not all simple points are selected as candidates. We only choose simple points within the difference set $\text{Diff}(f_B, g) = f_B \oplus g$.

Second, since we want to minimize the difference between the warped and target masks, we propose to flip pixels within the difference region $\text{Diff}(f_B, g)$. To implement this strategy efficiently, we order all pixels within $\text{Diff}(f_B, g)$ according to their distance from the FG/BG, and flip them according to this order. A pixel is skipped if it is not simple.

Our algorithm is based on the intuition that a far-away pixel will not become simple until nearby pixels are flipped first. To see this, we first formalize the definition of *distance transform* from the masks, f_B and g, denoted by D^{f_B} and D^g. For a BG pixel of g, p, its distance value $D^g(p)$ is the shortest distance from p to any FG pixel of g, $D^g(p) = \min_{s \in FG_g} \text{dist}(p, s)$. Similarly, for a FG pixel of g, q, $D^g(q) = \min_{s \in BG_g} \text{dist}(q, s)$. The definition generalizes to D^{f_B}.

We observe that a pixel cannot be simple unless it has distance 1 from the FG/BG of a warping mask. The proof is straightforward. Formally,

Lemma 1. *Given a 2D binary mask m, a pixel p cannot be simple for m if its distance function $D^m(p) > 1$.*

Proof. Assume the foreground has a pixel value of 1 and p is a background pixel with a index of (i, j). Consider the m-adjacent (m=4) for p. Since $D^m(p) > 1$, then we have $m(i-1, j) = m(i+1, j) = m(i, j-1) = m(i, j+1) = 0$. In this case, p is not 4-adjacent to any FG connected component, violating the **1)** of **Definition 1**. Consequently, pixel (i, j) is not a simple point. This also holds for foreground pixels. This lemma naturally generalizes to 3D case. □

Lemma 1 implies that only after flipping pixels with distance 1, the other misclassified locations should be considered. This observation gives us the intuition of our algorithm. To warp g towards f_B, our algorithm is as follows: (1) compute the difference set $\text{Diff}(f_B, g)$ as the candidate set of pixels; (2) sort candidate pixels in a non-decreasing order of the distance transform D^g; (3) enumerate through all candidate pixels according to the order. For each iteration, check if it is simple. If yes, flip the pixel's label. It is possible that this algorithm can miss some pixels.

They are not simple when the algorithm checks, but they might become simple as the algorithm continues (since their neighboring pixels get flipped in previous iterations).

One remedy is to recalculate the distance transform after one round of warping and go through the remaining pixels once more. But in practice, we found this is not necessary as this scenario is very rare.

By using Distance-Ordered Homotopy Warping, we are able to only consider all the inconsistent pixels once and flip them if they are simple. Otherwise, we need to iteratively warp all the inconsistent pixels (one non-simple pixel might become simple if its neighbors are flipped in the previous iteration). It takes 1.452s to warp a 512×512 image without the warping strategy, while only 0.317s with the strategy thereby allowing the network to converge much faster.

We conduct extensive experiments to demonstrate the effectiveness of the proposed method. Sec. 5.2.6 introduces the datasets used in this chapter, including both 2D and 3D datasets. The benchmark methods are described in Sec. 5.2.8. We mainly focus on topology-aware segmentation methods. Sec. 5.2.7 describes the evaluation metrics used to assess the quality of the segmentation. To demonstrate the ability to achieve better structural/topological performances, besides standard segmentation metrics, such as the DICE score, we also use several topology-aware metrics to evaluate all the methods. Several ablation studies are then conducted to further demonstrate the efficiency and effectiveness of the technical contributions (Sec. 5.3.1).

5.2.6 Datasets

We conduct extensive experiments to validate the efficacy of our method. Specifically, we use four natural and biomedical 2D datasets (**RoadTracer** [194], **DeepGlobe** [195], **Mass.** [171], **DRIVE** [172]) and one more 3D biomedical dataset (**CREMI** [2]) to validate the efficacy of the propose method. **Mass.** [171], **DRIVE** and **CREMI** have been introduce in Sec. 3.3.1. And the details of the other two datasets are listed as follows:

- **RoadTracer**: Roadtracer contains 300 high resolution satellite images, covering urban areas of forty cities from six different countries [194]. Similar to setting in [194], twenty five cities (180 images) are used as the training set, and the rest fifteen cities (120 images) are used as the validation set.

[2] https://cremi.org/

- **DeepGlobe**: DeepGlobe contains aerial images of rural areas in Thailand, Indonesia, and India [195]. Similar to setting in [179], we use 4696 images as the training set and the rest 1530 images as the validation set.

5.2.7 Evaluation Metrics

We use both pixel-wise (**DICE**) and topology-aware metrics (**ARI**, **Warping Error** and **Betti number error**) to evaluate the performance of the proposed method. **ARI** and **Betti number error** have been introduced in Sec. 3.3.2. The details of the other metrics used in this chapter are listed as follows:

- *DICE*: DICE score (also known as DICE coefficient, DICE similarity index) is one of the most popular evaluation metrics for image segmentation, which measures the overlapping between the predicted and ground truth masks.
- *Warping Error* [189]: Warping Error is metric that measures topological disagreements instead of simple pixel disagreements. After warping all the simple points of ground truth to the predicted mask, the disagreements left are topological errors. The warping error is defined as the percentage of these topological errors over the image size.

5.2.8 Baselines

We compare the results of our method with several state-of-the-art methods. The standard/simple UNet (2D/3D) **UNet** [6, 196] is used as a strong baseline and the backbone for other methods. The other baselines used in this chapter include **RoadTracer** [194], **VecRoad** [197], **iCurb** [198], **DIVE** [1], **UNet-VGG** [94], **TopoLoss** [2], **clDice** [109] and **DMT** [8]. **UNet**, **DIVE** [1] and **UNet-VGG** [94] have been introduce in Sec. 3.3.3. The other baseline methods used in this chapter are listed as follows:

- **RoadTracer** [194]: RoadTracer is an iterative graph construction based method where node locations are selected by a CNN.
- **VecRoad** [197]: VecRoad is a point-based iterative graph exploration scheme with segmentation-cues guidance and flexible steps.
- **iCurb** [198]: iCurb is an imitation learning-based solution for an offline road-curb detection method.

- **TopoNet** [2]: TopoNet is a recent work that tries to learn to segment with correct topology based on a novel persistent homology based loss function.
- **clDice** [109]: Another topology aware method for tubular structure segmentation. The basic idea is to use thinning techniques to extract the skeletons (centerlines) of the likelihood maps and ground truth mask. A new cldice loss is proposed based on the extracted skeletons besides a traditional pixel-wise loss.
- **DMT** [8]: DMT is a topology-aware deep image segmentation method via discrete Morse theory. Instead of identifying topological critical pixels/locations, the DMT loss tries to identify the whole morse structures, and the new loss is defined on the identified morse structures.

Note that *RoadTracer*, *VecRoad*, and *iCurb* are graph-based methods for road tracing. Graph-based approaches learn to explicitly detect keypoints and connect them. Since the graph is built iteratively, some detection errors in the early stage can propagate and lead to even more errors. Segmentation-based methods avoid this issue as they make predictions in a global manner. The challenge with segmentation methods in road network reconstruction is they may fail in thin structures especially when the signal is weak. This is exactly what we are addressing in this chapter – using critical pixels to improve segmentation-based methods.

UNet-VGG, *TopoNet*, *clDice* and *DMT* are topology-aware segmentation methods.

5.2.9 Implementation Details

For 2D images, we use (m, n) = (4, 8) to check if a pixel is simple or not; while (m, n) = (8, 26) for 3D images. We choose cross-entropy loss as L_{warp}/L_{pixel} for all the experiments, except the ablation studies in Tab. 5.3.

For 2D datasets, the batch size is set as 16, and the initial learning rate is 0.01. We randomly crop patches with the size of 512×512 and then feed them into the 2D UNet. For 3D case, the batch size is also 16, while the input size is $128 \times 128 \times 16$. We perform the data normalization for the single patch based on its mean and standard deviation.

We use PyTorch framework (Version: 1.7.1) to implement the proposed method. A simple/standard UNet (2D or 3D) is used as the baseline and the backbone. For the proposed method, as well as the other loss function based baselines, to make a fair comparison, we use the same UNet as the backbone. And the training strategy

is to train the UNet with dice loss first until converges and then add the proposed losses to fine-tune the models obtained from the initial step.

5.2.10 Results

In Fig. 5.8, we show qualitative results from different datasets. Compared with baseline UNet, our method recovers better structures, such as connections, which are highlighted by red circles. Our final loss is a weighted combination of the dice loss and warping-loss term L_{warp}. When $\lambda_{warp} = 0$, the proposed method degrades to a standard UNet. The recovered better structures (UNet and *Warping* columns in Fig. 5.8) demonstrate that our warping-loss helps the deep neural networks to achieve better topological segmentations.

5.3 Experiments

Tab. 5.1 shows quantitative results for three 2D image datasets, RoadTracer, DeepGlobe, and The Massachusetts dataset, and one 3D image dataset, CREMI. The best performances are highlighted in bold. The proposed warping-loss usually achieves the best performances in both DICE score and topological accuracy (ARI, Warping Error, and Betti Error) over other topology-aware segmentation baselines.

5.3.1 Ablation Studies

To further explore the technical contributions of the proposed method and provide a rough guideline of how to choose the hyperparameters, we conduct several ablation studies. Note that all the ablation studies are conducted on the RoadTracer dataset.

The Impact of the Loss Weights. As seen in Eq. 5.4, our final loss function is a combination of dice loss and the proposed warping loss L_{warp}. The balanced term λ_{warp} controls the influence of the warping loss term, and it's a dataset dependent hyper-parameter. The quantitative results for different choices of λ_{warp} are illustrated in Tab. 5.2. For the RoadTracer dataset, the optimal value is 1×10^{-4}. From Tab. 5.2, we can find that different choices of λ_{warp} do affect the performances. The reason is that, if λ_{warp} is too small, the effect of the warping loss term is negligible. However, if λ_{warp} is too large, the warping loss term will compete with the L_{dice} and decrease the performance of the other easy-classified pixels. Note that within a reasonable range of λ_{warp}, all the choices contribute to

better performances compared to baseline (row '0', standard UNet), demonstrating the effectiveness of the proposed loss term.

The Choice of Loss Functions. The proposed warping loss is defined on the identified topological critical pixels. Consequently, any pixel-wise loss functions can be used to define the warping loss L_{warp}/L_{pixel}. In this section, we investigate how the choices of loss functions affect the performances. The quantitative results are shown in Tab. 5.3. Compared with mean-square-error loss (MSE) or Dice loss, the cross-entropy loss (CE) achieves the best performances in terms of topological metrics. On the other hand, all these three choices perform better than the baseline method (row 'w/o', standard UNet), which further demonstrates the contribution of the proposed loss term.

Comparison of Different Critical Pixel Selection Strategies. We also conduct additional experiments to demonstrate the effectiveness of the proposed critical pixel selection strategy. The first variation is flipping the simple pixels but removing the heuristic that uses the distance transform, and then using the remaining non-simple pixels as our critical pixel set, which is denoted as 'w/o' in Tab. 5.4. The other two variations (warping only in one direction: ground truth to prediction or prediction to ground truth) achieve reasonably better while still slightly inferior results to the proposed version. The reason might be that the proposed critical point selection strategy contains more complete topologically challenging pixels. Both ablation studies demonstrate the effectiveness of the proposed critical point selection strategy.

Comparison with Morphology Post-processing. To verify the necessity of the proposed method, we also compare the proposed method with traditional morphology post-processing, i.e., dilation, and erosion. Dilation and erosion are global operations. Though closing operation (dilates image and then erodes dilated image) could bridge some specific gaps/broken connections, it will damage the global structures. In practice, the gaps/broken will usually be more than a few pixels. If the kernel size is too small, the closing operation (dilate then erode) will hardly affect the final performance; while too big kernel sizes will join the separated regions. Tab. 5.5 lists post-processing results on baseline UNet.

The Efficiency of the Proposed Loss. In this section, we'd like to investigate the efficiency of the proposed method. Our warping algorithm contains two parts, the distance transform and the sorting of the distance matrix. The complexity for distance transform is $O(n)$ for a 2D image where n is the size of the 2D image, and $n = H \times W$. H, W are the height and width of the 2D image, respectively. And the complexity for sorting is $mlog(m)$, where m is the number of misclassified pixels. Usually, $m \ll n$, so the overall complexity for the warping algorithm is

$O(n)$. As a comparison, [2] needs $O(n^3)$ complexity to compute the persistence diagram. And the computational complexity for [8], [109] are $O(nlog(n))$ and $O(n)$, respectively.

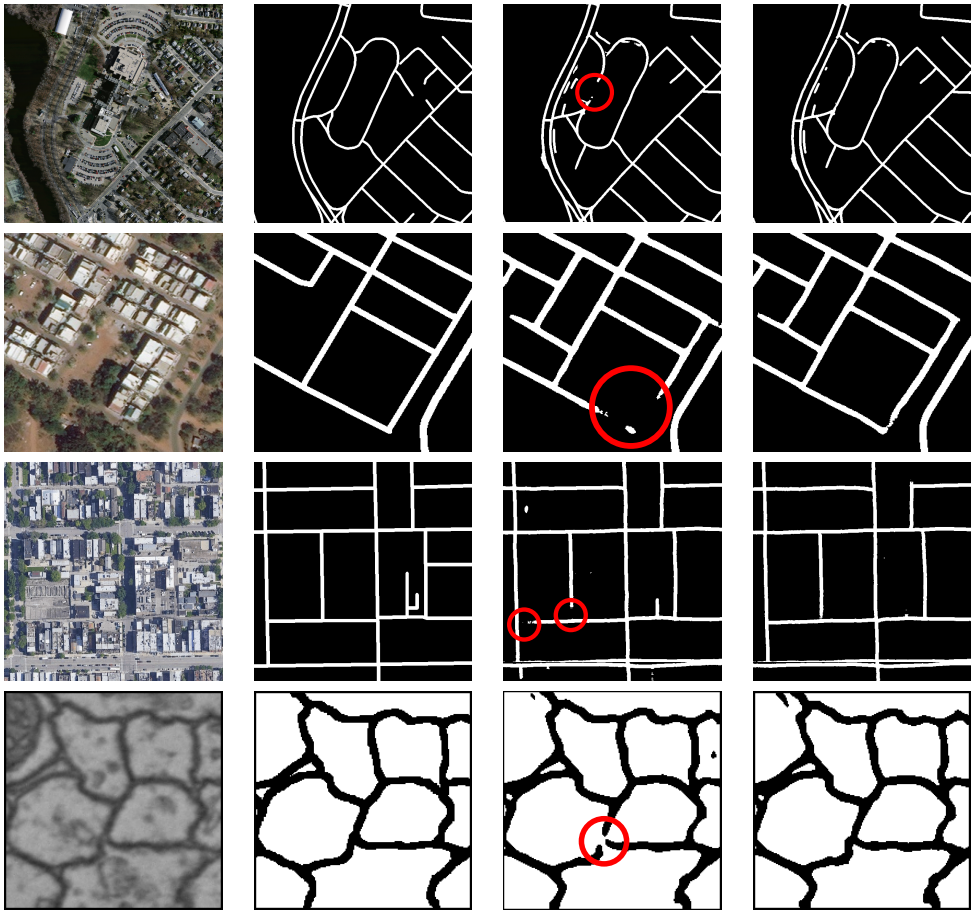
The comparison in terms of complexity and training time are illustrated in Tab. 5.6. Note that for the proposed method and all the other baselines, we first train a simple/standard UNet, and then add the additional loss terms to fine-tune the models obtained from the initial step. Here, the training time is only for the fine-tune step. The proposed method takes slightly longer training time than clDice, while achieving the best performance over the others. As all these methods use the same backbone, the inference times are the same.

5.3.2 Failure Cases

In this section, we add a few failure cases from different datasets. Note that inferring topology given an image is a very difficult task, especially near challenging spots, e.g., blurred membrane locations or weak vessel connections. Current methods can help to improve topology-wise accuracy, but they are far from perfect.

5.4 Conclusion

In this chapter, we propose a novel homotopy warping loss to learn to segment with better structural/topological accuracy. Under the homotopy warping strategy, we can identify the topological critical pixels/locations, and the new loss is defined on these identified pixels/locations. Furthermore, we propose a novel strategy called Distance-Ordered Homotopy Warping to efficiently identify the topological error locations based on distance transform. Extensive experiments on multiple datasets and ablation studies have been conducted to demonstrate the efficacy of the proposed method.



(a) Original (b) GT (c) UNet (d) Ours

Figure 5.8: Qualitative results compared with the standard UNet. The proposed warping loss can help to correct the topological errors (highlighted by red circles). The sampled patches are from four different datasets.

Table 5.1: Quantitative results of different methods.

Method	DICE↑	ARI↑	Warping↓	Betti↓
RoadTracer				
UNet [6]	0.587	0.544	10.412 $\times 10^{-3}$	1.591
RoadTracer [194]	0.547	0.521	13.224$\times 10^{-3}$	2.218
VecRoad [197]	0.552	0.533	12.819$\times 10^{-3}$	2.095
iCurb [198]	0.571	0.535	11.683$\times 10^{-3}$	1.873
UNet-VGG [94]	0.576	0.536	11.231 $\times 10^{-3}$	1.607
TopoNet [2]	0.584	0.556	10.008 $\times 10^{-3}$	1.378
clDice [109]	0.591	0.550	9.192 $\times 10^{-3}$	1.309
DMT [8]	0.593	0.561	9.452 $\times 10^{-3}$	1.419
Warping	**0.603**	**0.572**	**8.853** $\times 10^{-3}$	**1.251**
DeepGlobe				
UNet [6]	0.764	0.758	3.212 $\times 10^{-3}$	0.827
UNet-VGG [94]	0.742	0.748	3.371 $\times 10^{-3}$	0.867
TopoNet [2]	0.765	0.763	2.908 $\times 10^{-3}$	0.695
clDice [109]	0.771	0.767	2.874 $\times 10^{-3}$	0.711
DMT [8]	0.769	0.772	2.751 $\times 10^{-3}$	0.609
Warping	**0.780**	**0.784**	**2.683**$\times 10^{-3}$	**0.569**
Mass.				
UNet [6]	0.661	0.819	3.093$\times 10^{-3}$	3.439
UNet-VGG [94]	0.667	0.846	3.185$\times 10^{-3}$	2.781
TopoNet [2]	0.690	0.867	2.871$\times 10^{-3}$	1.275
clDice [109]	0.682	0.862	2.552$\times 10^{-3}$	1.431
DMT [8]	0.706	**0.881**	2.631 $\times 10^{-3}$	0.995
Warping	**0.715**	0.864	**2.440**$\times 10^{-3}$	**0.974**
DRIVE				
UNet [6]	0.749	0.834	4.781$\times 10^{-3}$	3.643
DIVE [1]	0.754	0.841	4.913$\times 10^{-3}$	3.276
UNet-VGG [94]	0.721	0.887	4.362$\times 10^{-3}$	2.784
TopoNet [2]	0.761	0.902	3.895$\times 10^{-3}$	1.076
clDice [109]	0.753	0.896	4.012 $\times 10^{-3}$	1.218
DMT [8]	0.773	0.908	3.561 $\times 10^{-3}$	0.873
Warping	**0.781**	**0.911**	**3.419**$\times 10^{-3}$	**0.812**
CREMI				
3D UNet [196]	0.961	0.832	11.173 $\times 10^{-3}$	2.313
DIVE [1]	0.964	0.851	11.219 $\times 10^{-3}$	2.674
TopoNet [2]	0.967	0.872	10.454 $\times 10^{-3}$	1.076
clDice [109]	0.965	0.845	10.576 $\times 10^{-3}$	0.756
DMT [8]	**0.973**	0.901	10.318 $\times 10^{-3}$	0.726
Warping	0.967	**0.907**	**9.854** $\times 10^{-3}$	**0.711**

Table 5.2: Ablation study for loss weight λ_{warp}.

λ_{warp}	DICE↑	ARI↑	Warping↓	Betti↓
0	0.587	0.544	10.412 $\times 10^{-3}$	1.591
2×10^{-5}	**0.603**	0.561	9.012 $\times 10^{-3}$	1.307
5×10^{-5}	0.601	0.548	9.356 $\times 10^{-3}$	1.412
1×10^{-4}	**0.603**	**0.572**	**8.853** $\times 10^{-3}$	**1.251**
2×10^{-4}	0.602	0.565	9.131 $\times 10^{-3}$	1.354

Table 5.3: Ablation study for the choices of loss.

L_{pixel}	DICE↑	ARI↑	Warping↓	Betti↓
w/o	0.587	0.554	10.412 $\times 10^{-3}$	1.591
MSE	0.598	0.556	9.853 $\times 10^{-3}$	1.429
Dice loss	**0.606**	0.563	9.471 $\times 10^{-3}$	1.368
CE	0.603	**0.572**	**8.853** $\times 10^{-3}$	**1.251**

Table 5.4: Comparison of different critical pixel selection strategies.

Kernel Size	DICE↑	ARI↑	Warping ↓	Betti↓
UNet	0.587	0.544	10.412 $\times 10^{-3}$	1.591
w/o DT	0.586	0.547	10.256$\times 10^{-3}$	1.473
Warping (GT $\rightarrow$ Pred)	0.594	0.567	9.171$\times 10^{-3}$	1.290
Warping (Pred $\rightarrow$ GT)	0.598	0.562	9.124 $\times 10^{-3}$	1.315
Warping	**0.603**	**0.572**	**8.853**$\times 10^{-3}$	**1.251**

Table 5.5: Comparison against post-processing.

Kernel Size	DICE↑	ARI↑	Warping ↓	Betti↓
UNet	0.587	0.544	10.412×10^{-3}	1.591
Closing (5)	0.588	0.542	10.414×10^{-3}	1.590
Closing (10)	0.587	0.546	10.399×10^{-3}	1.583
Closing (15)	0.586	0.541	10.428×10^{-3}	1.598
Warping	**0.603**	**0.572**	**8.853**$\times 10^{-3}$	**1.251**

Table 5.6: Comparison of efficiency.

Method	Complexity	Training time
TopoNet [2]	$O(n^3)$	$\approx 12h$
clDice [109]	$O(n)$	$\approx$ **3h**
DMT [8]	$O(nlogn)$	$\approx 7h$
Warping	$O(n)$	$\approx 4h$

(a) Original (b) GT (c) Ours

Figure 5.9: A few failure cases of the proposed method. From top to bottom, the sampled patches are from RoadTracer, DeepGlobe, Mass, DRIVE, and CREMI datasets respectively.

Chapter 6

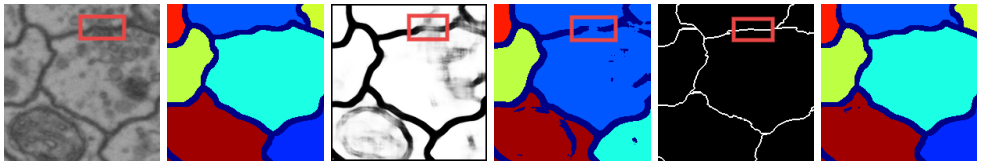
Topology-Aware Segmentation Using Discrete Morse Theory

In both Chapter 3 and Chapter 5, we introduce methods identifying a set of critical points of the likelihood function, e.g., saddles and extrema, as topologically critical locations for the neural network to memorize. However, only identifying a sparse set of critical points at every epoch is inefficient in terms of training. In this chapter, we introduce a novel method to identify critical structures instead of critical points.

6.1 Introduction

Segmenting objects while preserving their global structure is a challenging yet important problem. Various methods have been proposed to encourage neural networks to preserve fine details of objects [10–14]. Despite their high per-pixel accuracy, most of them are still prone to structural errors, such as missing small object instances, breaking thin connections, and leaving holes in membranes. These structural errors can significantly damage downstream analysis. For example, in the segmentation of biomedical structures such as membranes and vessels, small pixel errors at a junction will induce significant structure errors, leading to catastrophic functional mistakes. See Fig. 6.1 for an illustration.

Topology is a very global characterization that needs a lot of observations to learn. Any training set is insufficient in teaching the network to correctly reason about topology, especially near challenging spots, e.g., blurred membrane locations or weak vessel connections. A neural network tends to learn from clean-cut cases and converge quickly. Meanwhile, topologically-challenging locations remain



(a) (b) (c) (d) (e) (f)

Figure 6.1: Illustration of the importance of topological correctness in a neuron image segmentation task and the effectiveness of the proposed DMT-loss. The goal of this task is to segment membranes that partition the image into regions corresponding to neurons. **(a)** an input neuron image with challenging locations (blur regions) highlighted. **(b)** ground truth segmentation of the membranes (dark blue) and the result neuron regions. **(c)** likelihood map of a baseline method without topological guarantee [6]. **(d)** segmentation results of the baseline method. Small pixel-wise errors lead to broken membranes, resulting in the merging of many neurons into one. **(e)** The topologically critical structure captured by the proposed DMT-loss (based on the likelihood in (c)). **(f)** Our method produces the correct topology and the correct partitioning of neurons.

misclassified, causing structural/topological errors. We note that this issue cannot be alleviated even with more annotated (yet still unbalanced) images.

We propose a novel approach that identifies critical topological structures during training and teaches a neural network to learn from these structures. Our method can produce segmentations with correct topology, i.e., having the same *Betti number* (i.e., number of connected components and handles/tunnels) as the ground truth. Underlying our method is the classic Morse theory [16], which captures singularities of the gradient vector field of the likelihood function. Intuitively speaking, we treat the likelihood as a terrain function and Morse theory helps us capture terrain structures such as ridges and valleys. These structures, composed of 1D and 2D manifold pieces, reveal the topological information captured by the (potentially noisy) likelihood function.

We consider these Morse structures as *topologically critical*; they encompass all potential skeletons of the object. We propose a new loss that identifies these structures and enforce higher penalty along them. This way, we effectively address the sampling bias issue and ensure that the networks predict correctly near these topologically difficult locations. Since the Morse structures are identified based on the (potentially noisy) likelihood function, they can be both false negatives (a structure can be a true structure but was missed in the segmentation) and false

positives (a hallucination of the model and should be removed). Our loss ensures that both kinds of structural mistakes are corrected.

Several technical challenges need to be addressed. First, classical Morse theory was defined for smooth functions on continuous domains. Computing the Morse structures can be expensive and numerically unstable. Furthermore, the entire set of Morse structures may include an excessive amount of structures, a large portion of which can be noisy, irrelevant ones. To address these challenges, we use the discrete version of Morse theory by [17, 199]. For efficiency purposes, we also use an approximation algorithm to compute 2D Morse structures with almost linear time. The idea is to compute zero dimensional Morse structures of the dual image, which boils down to a minimum spanning tree computation. Finally, we use the theory of persistent homology [19, 20] to prune spurious Morse structures that are not relevant.

Our discrete-Morse-theory based loss called the *DMT-loss*, can be evaluated efficiently and can effectively train the neural network to achieve high performance in both topological accuracy and per-pixel accuracy. Our method outperforms state-of-the-art methods in multiple topology-relevant metrics (e.g., ARI and VOI) on various 2D and 3D benchmarks. It has superior performance in the Betti number error, which is an exact measurement of the topological fidelity of the segmentation.

6.2 Method

We propose a novel loss to train a topology-aware network end-to-end. It uses global structures captured by discrete Morse theory (DMT) to discover critical topological structures. In particular, through the language of 1- and 2-stable manifolds, DMT helps identify 1D skeletons or 2D sheets (separating 3D regions) that may be critical for structural accuracy. These Morse structures are used to define a DMT-loss that is essentially the cross-entropy loss constrained to these topologically critical structures. As the training continues, the neural network learns to better predict around these critical structures and eventually achieves better topological accuracy. Please refer to Fig. 6.2 for an overview of our method.

6.2.1 Morse Theory

Morse theory [16] identifies topologically critical structures from a likelihood map (Fig. 6.3(a)). In particular, it views the likelihood as a terrain function (Fig. 6.3(b)) and extracts its landscape features such as mountain ridges and their

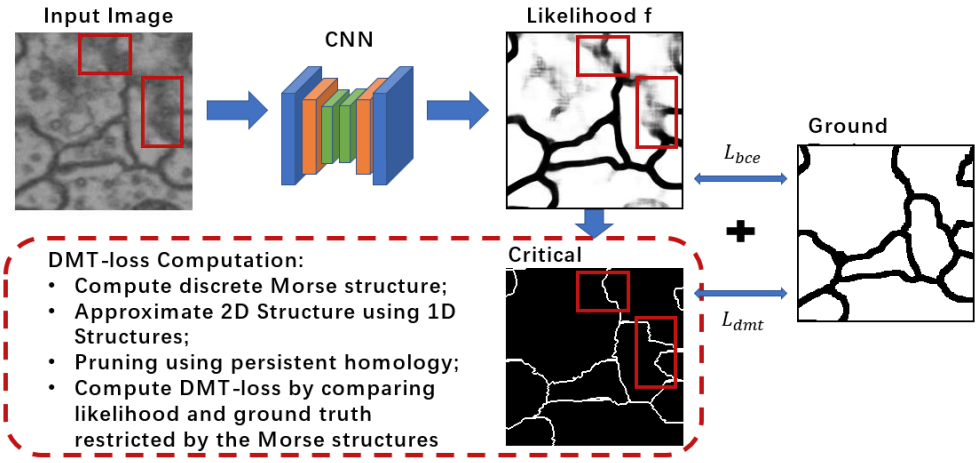
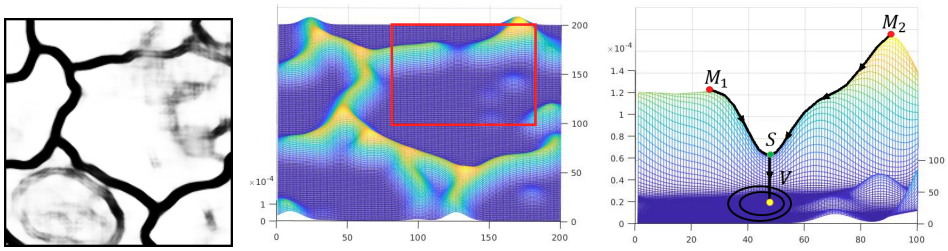


Figure 6.2: Overview of our method. Topologically critical and error-prune structures are highlighted.

high-dimensional counterparts. The broken connection in the likelihood map corresponds to a local dip in the mountain ridge of the terrain in Fig. 6.3(b) and Fig. 6.3(c). The bottom of this dip is captured by a so-called saddle point (S in Fig. 6.3(c)) of the likelihood map. The mountain ridge connected to this bottom point captures the main part of the missing pixels. Such "mountain ridges" can be captured by the so-called stable manifold w.r.t. the saddle point using the language of Morse theory. By finding the saddle points and the stable manifold of the saddle points on the likelihood map, we can ensure the model learns to "correctly" handle pixels near these structures. We note that an analogous scenario can also happen with such 1D signals (such as blood vessels) as well as 2D signals (such as membranes of cells) in 3D images – they can also be captured by saddles (of different indices) and their stable manifolds.

In this chapter, we focus on the application of segmenting 2D and 3D images. Specifically, suppose we have a smooth function $f : \mathbb{R}^d \to \mathbb{R}$ to model the likelihood (density) map. Given any point $x \in \mathbb{R}^d$, the negative gradient $-\nabla f(x) = -[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_d}]^T$ indicates the steepest descending direction of f. A point $x = (x_1, x_2, \ldots, x_k)$ is *critical* if the function gradient at this point vanishes (i.e., $\nabla f(x) = 0$). For a well-behaved function (more formally, called Morse function) defined on $\mathbb{R}^d$, a critical point could be a minimum, a maximum, or $d-1$ type of saddle point. See Fig. 6.3(c) for an example. For $d = 2$, there is only one saddle point type. For $d = 3$, there are two saddle point types, referred to as index-1 and index-2 saddles. Formally, when taking the eigenvalues of the



(a) (b) (c)

Figure 6.3: From left to right: **(a)** Likelihood map. **(b)** Density map: the z-axis value is the probability of the likelihood map in the left figure. **(c)** Density map for the highlighted region in the middle figure. M_1 and M_2 are maxima (red dots), V is a minimum (yellow), S is a saddle (green) with its stable manifolds flowing to it from M_1 and M_2.

Hessian matrix at a critical point, its index is equal to the number of negative eigenvalues.

Intuitively, imagine we put a drop of water on the graph of f (i.e, the terrain in Fig. 6.3(b)) at the lift of x onto this terrain, then $-\nabla f(x)$ indicates the direction along which the water will flow down. If we track the trajectory of this water drop as it flows down, this gives rise to a so-called *integral line* (a flow line). Such flow lines can only start and end at critical points [1], where the gradient vanishes.

The stable manifold $\mathsf{S}(p)$ of a critical point p is defined as the collection of points whose flow line ends at p. For a 2D function $f : \mathbb{R}^2 \to \mathbb{R}$, for a saddle q, its stable manifold $\mathsf{S}(q)$ starts from local maxima (mountain peaks in the terrain) and ends at q, tracing out the mountain ridges separating different valleys (Fig. 6.3(c)). The stable manifold $\mathsf{S}(p)$ of a minimum p, on the other hand, corresponds to the entire valley around this minimum p. See the valley point V and its corresponding stable field in Fig. 6.3(c). For a 3D function $f : \mathbb{R}^3 \to \mathbb{R}$, the stable manifold w.r.t. an index-2 saddle connects mountain peaks to saddles, tracing 1D mountain ridges as in the case of a 2D function. The stable manifold w.r.t. an index-1 saddle q consists of flow lines starting at index-2 saddles and ending at q. Their union, called the 2-stable manifold of f, consists of a collection of 2-manifold pieces.

These stable manifolds indicate important topological structures (graph-like or sheet-like) based on the likelihood of the current neural network. Using these structures, we will propose a novel loss (Sec. 6.2.3) to improve the topological

[1] More precisely, flow lines only tend to critical points in the limit and never reach them.

awareness of the model. In practice, for images, we will leverage the discrete version of Morse theory for both numerical stability and easier simplification.

Discrete Morse Theory. We view a dD image, $d = 2$ or 3, as a d-dimensional cubical complex, meaning it consists of 0-, 1-, 2- and 3-dimensional cells corresponding to vertices, edges, squares, and voxels (cubes) as its building blocks.

Discrete Morse theory (DMT), originally introduced in [17, 199], is a combinatorial version of Morse theory for general cell complexes. There are many beautiful results established for DMT, analogous to classical Morse theory. We will however only briefly introduce some relevant concepts for the present chapter, and we will describe them in the setting of cubical complexes (instead of simplicial complexes) as they are more suitable for images.

Let K be a cubical complex. Given a p-cell τ, we denote by $\sigma < \tau$ if σ is a $(p-1)$-dimensional face for τ. A *discrete gradient vector* (also called *a V-pair* for simplicity) is a pair (τ, σ) where $\sigma < \tau$. Now suppose we are given a collection of V-pairs $\mathsf{M}(K)$ over the cubical complex K. A sequence of cells $\pi : \tau_0^{p+1}, \sigma_1^p, \tau_1^{p+1}, \sigma_2^p, \cdots, \sigma_k^p, \tau_k^{p+1}, \sigma_{k+1}^p$, where the superscript p in α^p stands for the dimension of this cell α, form a *V-path* if $(\tau_i, \sigma_i) \in \mathsf{M}(K)$ for for any $i \in [1, k]$ and $\sigma_i < \tau_{i-1}$ for any $i \in [1, k+1]$. A V-path π is *acyclic* if $(\tau_0, \sigma_{k+1}) \notin \mathsf{M}(K)$. This collection of V-pairs $\mathsf{M}(K)$ form a *discrete gradient vector field* [2] if (cond-i) each cell in $\mathsf{M}(K)$ can only appear in at most one pair in $\mathsf{M}(K)$; and (cond-ii) all V-paths in $\mathsf{M}(K)$ are acyclic. Given a discrete gradient vector field $\mathsf{M}(K)$, a simplex $\sigma \in K$ is *critical* if it is not in any V-pair in $\mathsf{M}(K)$.

Even though a discrete gradient vector (a V-pair), say (τ, σ) is a combinatorial pair instead of a real vector, it still indicates a "flow" from τ to its face σ. A V-path thus corresponds to a flow path (integral line) in the smooth setting. However, to make a collection of V-pairs a valid analog of gradient field, (cond-i) says that at each simplex there should only be one "flow" direction; while (cond-ii) is necessary as flow lines traced by gradient can only go down in function values and thus never come back (thus acyclic).

A critical simplex has a "vanishing gradient" as it is not involved in any V-pair in $\mathsf{M}(K)$ (i.e, there is no flow at this simplex). Given a 2D cubical complex K a discrete gradient vector field $\mathsf{M}(K)$, we can view critical 0-, 1-, 2- and 3-cells as minima, saddle points, and maxima, respectively. If K is 3D, then we can view critical 0-, 1-, 2- and 3-cells as minima, index-1 saddle, index-2 saddle and maxima, respectively.

[2] We will not introduce the concept of discrete Morse function, as the discrete gradient vector field is sufficient to define all relevant notations.

Hence, a 1-stable manifold in 2D will correspond to a V-path connecting a critical square (a maximum) and a critical edge (a saddle), while in 3D, it will be a V-path connecting a critical cube and a critical square.

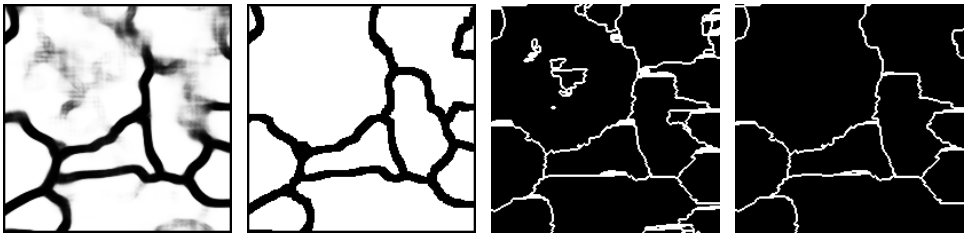
Morse Cancellation. A given discrete gradient field $\mathsf{M}(K)$ could be noisy, e.g, there are shallow valleys where the mountain ridge around it should be ignored. Fortunately, the discrete Morse theory provides an elegant and purely combinatorial way to cancel pairs of critical simplices (and thus reduce their stable manifolds). In particular, given $\mathsf{M}(K)$, a pair of critical simplices $\langle \delta^{(p+1),\gamma^p} \rangle$ is *cancellable* if there is a unique V-path $\pi = \delta = \delta_0, \gamma_1, \delta_1, \ldots, \delta_s, \gamma_{s+1} = \gamma$ from δ to γ. The *Morse cancellation operation* simply reverses all V-pairs along this path by removing all V-pairs along this path and adding (δ_{i-1}, γ_i) to $\mathsf{M}(K)$ for any $i \in [1, s+1]$. It is easy to check that after the cancellation neither δ nor γ is critical.

6.2.2 Simplification and Computation

In this section, we describe how we extract discrete Morse structures corresponding to the 1-stable and 2-stable manifolds in the continuous analog. First, we prune unnecessary Morse structures, based on the theory of persistent homology [19, 20]. Second, we approximate the 2-stable manifold structures using 0-stable manifolds of the dual to achieve high efficiency in practice, because it is rather involved to compute based on the original definition.

Persistence-based Structure Pruning. While Morse structures reveal important structural information, they can be sensitive to noise. Without proper pruning, there can be an excessive amount of Morse structures, many of which are spurious and not relevant to the true signal. See Fig. 6.4(c) for an example. Similar to previous approaches e.g., [118, 120, 200], we will prune these structures using persistent homology.

Persistent homology is one of the most important developments in the field of topological data analysis in the past two decades [19, 20]. Intuitively speaking, we grow the complex by starting from the empty set and gradually including more and more cells using a decreasing threshold. Through this course, new topological features can be created upon adding a critical cell, and sometimes a feature can be destroyed upon adding another critical cell. The persistence algorithm [20] will pair up these critical cells; that is, its output is a set of critical cell pairs, where each pair captures the birth and death of topological features during this evolution. The persistence of a pair is defined as the difference of function values of the two critical cells, intuitively measuring how long the topological feature lives in terms of f.



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

Figure 6.4: From left to right: **(a)** Sample likelihood map, **(b)** Ground truth, **(c)** improperly pruned structures and **(d)** properly pruned structures.

Using persistence, we can prune critical cells that are less topologically salient, and thus their corresponding Morse structures. Recall each 1- and 2-stable Morse structure is constituted by V-paths flowing into a critical cell (corresponding to a saddle in the continuous setting). We then use the persistence associated with this critical cell to determine the saliency of the corresponding Morse structure. If the persistence is below a certain threshold ϵ, we prune the corresponding Morse structure via an operation called *Morse cancellation*. [3] See Fig. 6.4(d) for example Morse structures after pruning. We denote by $\mathcal{S}_1(\epsilon)$ and $\mathcal{S}_2(\epsilon)$ the remaining sets of 1- and 2-stable manifolds after pruning. We'll use these Morse structures to define the loss (Sec. 6.2.3).

Persistence Pruning. We can extend this vertex-valued function ρ to a function $\rho : K \to \mathbb{R}$, by setting $\rho(\sigma)$ for each cell to be the maximum ρ-value of each vertex in σ. How to obtain a discrete gradient vector field from such function $\rho : K \to \mathbb{R}$? Following the approach developed in [120, 201], we initialize a trivial discrete gradient vector field where all cells are initially critical. Let $\epsilon > 0$ be a threshold for simplification. We then perform persistence algorithm [20] induced by the super-level set filtration of ρ and pair up all cells in K, denoted by $\mathcal{P}_\rho(K)$.

Persistent homology is one of the most important developments in the field of topological data analysis in the past two decades [19,20,160]. We will not introduce it formally here. Imagine we grow the complex K by starting from the empty set and gradually including more and more cells in decreasing ρ values. (More formally, this is the so-called super-level set filtration of K induced by ρ.) Through this course, the new topological features can be created upon adding a simplex σ,

[3] Technically, not all spurious structures can be pruned/canceled. But in practice, most of them can.

and sometimes a feature can be destroyed upon adding a simplex τ. Persistence algorithm [20] will pair up simplices; that is, its output is a set of pairs of simplices $\mathcal{P}_\rho(K) = \{(\sigma, \tau)\}$, where each pair captures the birth and death of topological features during this evolution. The persistence of a pair, say $\mathsf{p} = (\sigma, \tau)$, is defined as $\mathrm{pers}(\mathsf{p}) = \rho(\sigma) - \rho(\tau)$, measuring how long the topological feature captured by p lives in term of ρ. In this case, we also write $\mathrm{pers}(\sigma) = \mathrm{pers}(\tau) = \mathrm{pers}(\mathsf{p})$ – the persistence of a simplex (say σ or τ) can be viewed as the importance of this simplex.

With this intuition of the persistence pairings, we next perform Morse cancellation operation to all pairs of these cells $(\sigma, \tau) \in \mathcal{P}_\rho(K)$ in increasing order their persistence if (i) its persistence $\mathrm{pers}(\delta, \gamma) < \epsilon$ (i.e, this pair has low persistence and thus not important); and (ii) this pair (δ, γ) is cancellable.

Let $\mathsf{M}_\epsilon(K)$ be the resulting discrete gradient field after simplifying all low-persistence critical simplices. We then construct the 1-stable and 2-stable manifolds for the remaining (high persistence, and thus important) saddles (critical 1-cell and 2-cells) from $\mathsf{M}_\epsilon(K)$. Let $\mathcal{S}_1(\epsilon)$ and $\mathcal{S}_2(\epsilon)$ be the resulting collection of 1- and 2-stable manifolds respectively. In particular, see an illustration of a V-path (highlighted in black) corresponding to a 1-stable manifold of the green saddle in Fig. 3(c).

Computation. We need an efficient algorithm to compute $\mathcal{S}_1(\epsilon)$ and $\mathcal{S}_2(\epsilon)$ from a given likelihood f, because this computation needs to be carried out at each epoch. It is significantly more involved to define and compute $\mathcal{S}_2(\epsilon)$ in the discrete Morse setting [118]. Furthermore, this also requires the computation of persistent homology up to 2-dimensions, which takes time $T = O(n^\omega)$ (where $\omega \approx 2.37$ is the exponent in the matrix multiplication time, i.e., the time to multiply two $n \times n$ matrices). To this end, we propose to approximate $\mathcal{S}_2(\epsilon)$ by $\widehat{\mathcal{S}}_2(\epsilon)$ which intuitively comes from the "boundary" of the stable manifold for minima. Note that in the smooth case for a function $f : \mathbb{R}^3 \to \mathbb{R}$, the closure of 2-stable manifolds exactly corresponds to the 2D sheets on the boundary of stable manifolds of minima. [4] This is both conceptually clear and also avoids the costly persistence computation. In particular, given a minimum q with persistence greater than the pruning threshold ϵ, the collections of V-paths ending at the minimum q form a spanning tree T_q. In fact, consider all minima $\{q_1, \ldots, q_\ell\}$ with persistence at least ϵ. $\{T_{q_i}\}$ form a maximum spanning forest among all edges with persistence value smaller than ϵ [201, 202]. Hence it can be computed easily in $O(n \log n)$ time, where n is the image size.

[4] This however is not always true for the discrete Morse setting.

More Details on the Approximation of $\mathcal{S}_2$ via $\hat{S}_2$. We approximate S_2 by taking the boundary of the stable manifold of the minima (basins/valleys in the terrain). This is like a watershed algorithm: growing the basins from all minima until they meet. The stable manifolds of the minima are approximated using spanning trees. This algorithm is inspired by the continuous analog for Morse functions.

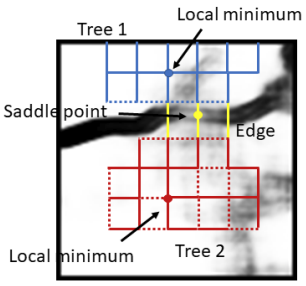


Figure 6.5: Spanning tree illustration.

We then take all the edges incident to nodes from different trees. The dual of these edges, denoted as $\widehat{\mathcal{S}}_2(\epsilon)$, serves as the "boundaries" separating different spanning trees (representing stable manifolds to different minima with persistence $\geq \epsilon$). See Fig. 6.5. Overall, the computation of $\widehat{\mathcal{S}}_2(\epsilon)$ takes only $O(n \log n)$ by a maximum spanning tree algorithm.

As for $\mathcal{S}_1(\epsilon)$, we use a simplified algorithm of [201], which can compute $\mathcal{S}_1(\epsilon)$ in $O(n \log n)$ time for a 2D image, in which n is the image size. For a 3D image, the time is $O(n \log n + T)$, where $T = O(n^\omega)$ is the time to compute persistent homology, where $\omega \approx 2.37$ is the exponent in matrix multiplication time.

6.2.3 The DMT-based Loss Function and Training Details

Our loss has two terms, the cross-entropy term, L_{bce} and the DMT-loss, L_{dmt}: $L(f, g) = L_{bce}(f, g) + \beta L_{dmt}(f, g)$, in which f is the likelihood, g is the ground truth, and β is the weight of L_{dmt}. Here we focus on one single image, while the actual loss is aggregated over the whole training set.

The DMT-loss enforces the correctness of the topologically challenging locations discovered by our algorithm. These locations are pixels of the (approximation of) 1- and 2-stable manifolds $\mathcal{S}_1(\epsilon)$ and $\widehat{\mathcal{S}}_2(\epsilon)$ of the likelihood, f. Denote by $\mathcal{M}_f$ a binary mask of the union of pixels of all Morse structures in $\mathcal{S}_1(\epsilon) \cup \widehat{\mathcal{S}}_2(\epsilon)$. We want to enforce these locations to be correctly segmented. We use the cross-entropy between the likelihood map f and ground truth g restricted to the Morse structures, formally, $L_{dmt}(f, g) = L_{bce}(f \circ \mathcal{M}_f, g \circ \mathcal{M}_f)$, in which $\circ$ is the Hadamard product.

Different Topological Error Types. Recall that the Morse structures are computed over the potentially noisy likelihood function of a neural network, which can help identify two types of structural errors: (1) **false negative**: a true structure that is incomplete in the segmentation, but can be visible in the Morse structures. These types of false negatives (broken connections, holes in membrane) can be restored

as the additional cross-entropy loss near the Morse structures will force the network to increase its likelihood value on these structures. (2) **false positive**: phantom structures hallucinated by the network when they do not exist (spurious branches, membrane pieces). These errors can be eliminated as the extra cross entropy loss on these structures will force the network to decrease the likelihood values along these structures. **Illustration of 3D Topological Errors.** We have already introduced discrete Morse theory with a 2D example. Here, we would like to illustrate 3D topological errors with 3D examples.

Fig. 6.6 and Fig. 6.7 illustrate two different types of topological errors for 3D data. Fig.6.6 illustrates an index-1 topological error for 3D synthetic data. 3D EM/neuron has the same type of topological error as the synthetic data. Fig.6.7 illustrates index-2 topological error for 3D vessel data.

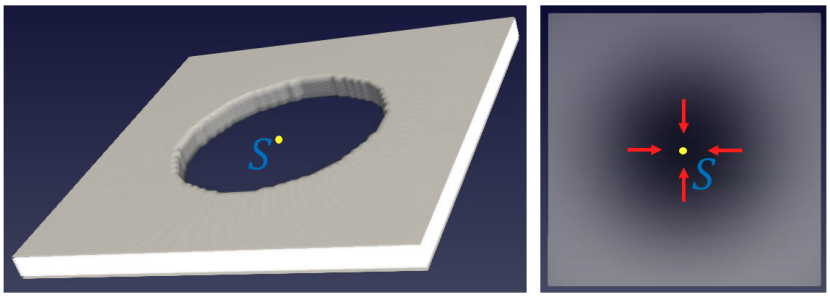


Figure 6.6: Illustration of an index-1 topological error (3D hole in the middle) for a 3D synthetic data. The ground truth is a complete sheet without a hole. We intentionally weaken the likelihood function in the middle. So the segmentation has a hole in the middle. **Left**: 3D segmentation result. S is a saddle point of the likelihood function. Its Hessian has 1 negative and 2 positive eigenvalues. The stable manifold of the saddle point S is a 2D plane going through the saddle point and cutting the segmentation into two thin slices. **Right**: the likelihood function visualized on the 2D stable manifold of S. Red arrows illustrate how different V-paths (streamlines of negative gradient) flow to the saddle S.

False Negative and False Positive Errors. We have mentioned that the proposed DMT-loss can capture and fix two different types of topological errors: false negative and false positive. We illustrate these two types in Fig. 6.8. The two highlighted red rectangles represent the two types of topological errors: 1) The red rectangle on the right represents a sample of false negative error; part of the membrane structure is missing, due to a blurred region near the membrane. 2) The

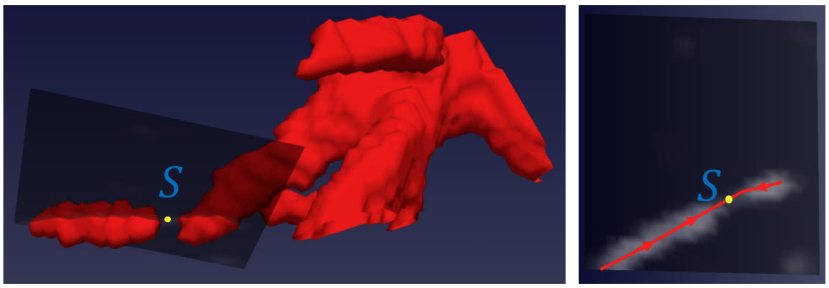


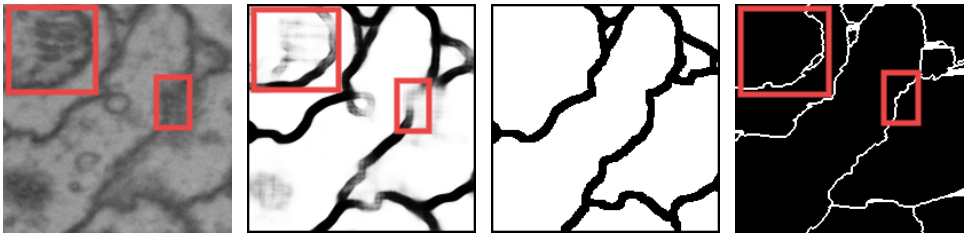
Figure 6.7: Illustration of an index-2 topological error from a 3D vessel image. The vessel segmentation has a broken connection near the bottom-left of the Left image. **Left**: part of the 3D segmentation result. The saddle point S corresponds to a broken connection. The Hessian of the likelihood at the saddle point has 2 negative and 1 positive eigenvalues. **Right**: One slice of the 3D likelihood map passing the saddle point. The saddle point (yellow) and its 1-stable manifold (red) are also drawn.

red rectangle on the left represents a sample of a false positive error. In this specific case, it is caused by mitochondria which are not the boundary of neurons.

In summary, with the help of the proposed DMT-loss, we can identify both these two types of topological errors, and then force the network to increase/decrease its likelihood value on these structures to correctly segment the images with the correct topology.

Differentiability. We note that the Morse structures are recomputed at every epoch. The structures, as well as their mask $\mathcal{M}_f$, may change with f. However, the change is not continuous; the output of the discrete Morse algorithm is a combinatorial solution that does not change continuously with f. Instead, it only changes at singularities, i.e., when the function values of f at different pixels/voxels are the same. In other words, for a general f, the likelihood function is real-valued, so it is unlikely two pixels share the exact same value. In case they are, the persistence homology algorithm by default will break the tie and choose one as critical. The mask $\mathcal{M}_f$ remains a constant within a small neighborhood of current f. Therefore, the gradient of L_{dmt} exists and can be computed naturally.

Training Details. Although our method is architecture-agnostic, for 2D datasets, we select an architecture driven by a 2D UNet [6]; for 3D datasets, we select an architecture inspired by a 3D UNet [196]. Both UNet and 3D UNet were originally designed for neuron segmentation tasks, capturing the fine-structures of images. In practice, we first pretrain the network with only the cross-entropy loss, and then



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

Figure 6.8: Illustration of two different types of topological errors captured by DMT-loss. **(a)** an input neuron image with challenging locations highlighted. **(b)** likelihood map of a baseline method without topological guarantee [6]. **(c)** ground truth. **(d)** The topologically critical structure from the likelihood, captured by the proposed discrete Morse algorithm. These structures will be used in the DMT-Loss.

train the network with the combined loss.

6.3 Experiments on 2D Datasets

6.3.1 Datasets

Six natural and biomedical 2D datasets are used: **ISBI12** [168], **ISBI13** [169], **CREMI**, **CrackTree** [170], **Mass.** [171] and **DRIVE** [172]. The details of the datasets have been introduced in Sec. 3.3.1. For all the experiments, we use a 3-fold cross-validation to tune hyperparameters for both the proposed method and other baselines, and report the mean performance over the validation set. This also holds for 3D experiments.

6.3.2 Evaluation Metrics

We use five different evaluation metrics: **Pixel-wise accuracy**, **DICE score**, **ARI**, **VOI**, and the most important one is **Betti number error**, which directly compares the topology (number of handles/voids) between the segmentation and the ground truth. More details about the evaluation metrics have been provided in Sec. 3.3.2 and Sec. 5.2.7. The last three metrics are topology-aware.

6.3.3 Baselines

We use **DIVE** [1], **UNet** [6], **UNet-VGG** [94] and **TopoLoss** [2] as baselines. All the baselines have been described in Sec. 3.3.3. For all methods, we generate segmentations by thresholding the predicted likelihood maps at 0.5, and this also holds for 3D experiments.

6.3.4 Results

Tab. 6.1 shows quantitative results for 2D image datasets. The results are highlighted when they are significantly better, and statistical significance is determined by t-tests. The DMT-loss outperforms others in both DICE score and topological accuracy (ARI, VOI, and Betti Error). Please note that here the backbone of TopoLoss is the same as in [2], a heavily engineered network. The performance of TopoLoss will be worse if we implement it using the same UNet backbone as DMT-Loss.

Fig. 6.9 shows qualitative results. Our method correctly segments fine structures such as membranes, roads, and vessels. Our loss is a weighted combination of the cross entropy and DMT-losses. When $\beta = 0$, the proposed method degrades to a standard UNet. The performance improvement over all datasets (UNet and DMT line in Tab. 6.1) demonstrates that our DMT-loss is helping the deep neural nets to learn a better structural segmentation.

Fairness Comparisons with Same Backbone Networks. We copy the numbers of TopoLoss from [2], which is TopoLoss+DIVE. And in this chapter, we use UNet as the backbone. Indeed, with UNet, TopoLoss will be worse and the gap will be even bigger. The DIVE used in [2] is more expensive and better designed specifically for EM images. We choose UNet in this manuscript as it is lightweight and easy to generalize to many datasets. We also apply our backbone-agnostic DMT-loss to the DIVE network [1]. All the experiments are conducted on CREMI 2D dataset. The quantitative results (Betti Error) are shown in Tab. 6.2.

6.4 Experiments on 3D Datasets

6.4.1 Datasets

We use three different biomedical 3D datasets: **ISBI13**, **CREMI** and **3Dircadb** [203]. **ISBI13** and **CREMI**, which have been discussed above, are originally 3D datasets, can be used in both 2D and 3D evaluations. We also evaluate our method on

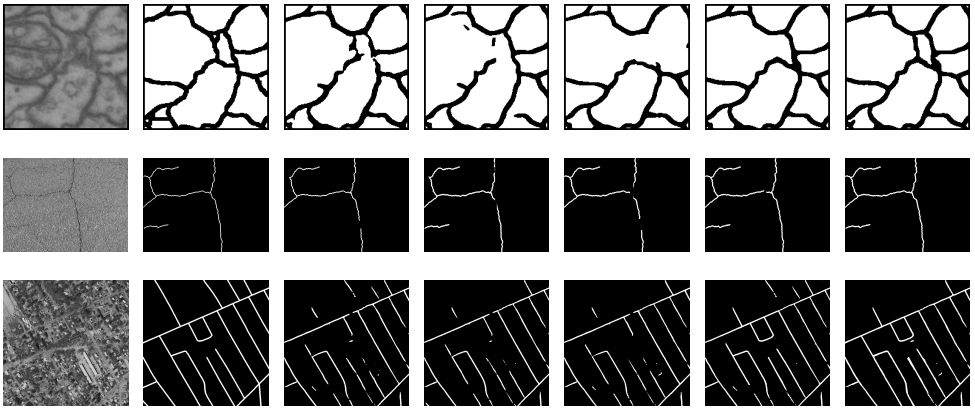


Figure 6.9: Qualitative results of the proposed method compared to other models. From left to right: sample images, ground truth, results for **DIVE**, **UNet**, **Mosin.**, **TopoLoss** and our proposed **DMT**.

the open dataset **3Dircadb**, which contains 20 enhanced CT volumes with artery annotations.

6.4.2 Evaluation Metrics

We use similar evaluation metrics as the 2D part. Note that, in terms of 2D images, for ARI and VOI, we compute the numbers for each slice and then average the numbers for different slices as the final performance; for 3D images, we compute the performance for the whole volume. For 2D images, we compute the 1D Betti number (number of holes) to obtain the Betti Error; while for 3D images, we compute the 2D Betti number (number of voids) to obtain the Betti Error.

6.4.3 Baselines

3D DIVE [204], **3D UNet** [196], **3D TopoLoss** [2] are the 3D versions for **DIVE**, **UNet** and **TopoLoss**. **MALA** [55] trains the UNet using a new structured loss function.

6.4.4 Results

Tab. 6.3 shows the quantitative results for three different 3D image datasets, ISBI13, CREMI, and 3Dircadb. Our method outperforms existing methods in topological accuracy (in all three topology-aware metrics), which demonstrates the effectiveness of the proposed method. Fig. 6.10 shows qualitative results for the ISBI13 dataset.

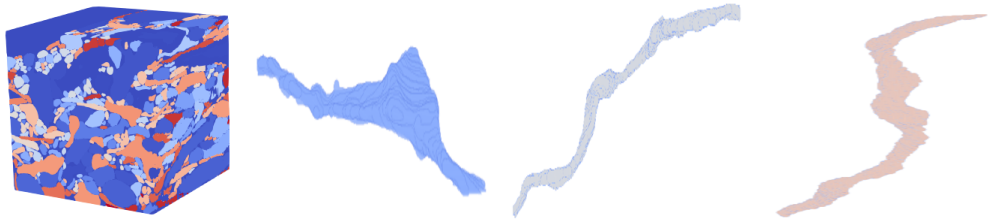


Figure 6.10: Segmentation results for ISBI13 dataset and 3 randomly selected neurons.

The Benefit of the Proposed DMT-loss. Instead of capturing isolated critical points in TopoLoss [2], the proposed DMT-loss captures the whole V-path as critical structures. Taking the patch in Fig. 6.4(a) as an example, TopoLoss identifies $\approx$ 80 isolated critical pixels for further training, whereas the critical structures captured by the DMT-loss contain $\approx$ 1000 critical pixels (Fig. 6.4(d)). We compare the efficiency of DMT-loss and TopoLoss using the same backbone network, evaluated on the CREMI 2D dataset. Both methods start from a reasonable pre-trained likelihood map. TopoLoss achieves 1.113 (Betti Error), taking $\approx$3h to converge; while DMT-loss achieves 0.956 (Betti Error), taking $\approx$1.2h to converge (the standard UNet takes $\approx$0.5h instead). Aside from converging faster, the DMT-loss is also less likely to converge to low-quality local minima. We hypothesize that the loss landscape of the topological loss will have more local minima than that of the DMT-loss, even though the global minima of both landscapes may have the same quality.

Ablation Study for Persistence Threshold ϵ. As illustrated in Fig. 6.4, different persistence thresholds ϵ will identify different critical structures. The ablation experiment is also conducted on the CREMI 2D dataset. When $\epsilon = 0.2$ (See Fig. 6.4(d)), the proposed DMT-loss achieves the best performance of 0.982 (Betti Error). When $\epsilon = 0.1$ and $\epsilon = 0.3$, the performance drops to 1.206 and 2.105 (both in Betti Error), respectively. This makes sense for the following reasons: 1) for

$\epsilon = 0.1$, the DMT-loss captures lots of unnecessary structures which mislead the neural networks; 2) for $\epsilon = 0.3$, the DMT-loss misses lots of important critical structures, making the performance drop significantly. The ϵ is chosen via cross-validation.

The Ablation Study for Balanced Term β. We conduct another ablation study for the balanced weight of parameter β. Note that, the parameter β is dataset dependent. We conduct the ablation experiment on CREMI 2D dataset. When $\beta = 3$, the proposed DMT-loss achieves the best performance of 0.982 (Betti Error). When $\beta = 2$ and $\beta = 4$, the performance drops to 1.074 and 1.181 (both in Betti Error), respectively. The parameter β is also chosen via cross-validation.

Comparison with Other Simpler Choices. The proposed method essentially highlights geometrically rich locations. To demonstrate the effectiveness of the proposed method, we also compare with two baselines: canny edge detection, and ridge detection, which achieve 2.971 and 2.507 in terms of Betti Error respectively, much worse than our results (Betti Error: 0.982). Although our focus is the Betti error, we also report per-pixel accuracy for reference (See Tab. 6.4 for details). From the results, we observe that the baseline models could not solve topological errors, even though they achieve high per-pixel accuracy. Without a persistent-homology based pruning, these baselines generate too many noisy structures and thus are not as effective as DMT-loss.

Robustness of the Proposed Method. We run another ablation study on images corrupted with Gaussian noise. The experiment is also conducted on the CREMI 2D dataset. From the results (See Tab. 6.5 for details), the DMT-loss is fairly robust and maintains good performance even with high noise levels. The reason is that the Morse structures are computed on the likelihood map, which is already robust to noise.

In Tab. 6.5, 10% means the percentage of corrupted pixels, and $\delta/2\delta$ means the sdv of the added Gaussian noise. For reference, we note that the performance of the standard UNet is 3.016 (Betti Error).

Comparison with Reweighted Cross Entropy Loss. We run an additional ablation study to compare with a baseline of reweighting the FP and FN pixels in the cross-entropy loss (Reweighting CE). The weights of the FP/FN pixels are hyperparameters tuned via cross-validation. The reweighting CE strategy achieves 2.753 in terms of Betti Error (on CREMI 2D data), and the DMT-loss is better than this baseline. The reason is that the DMT-loss specifically penalizes FP and FN pixels which are topology-critical. Meanwhile, reweighting CE adds weights to FP/FN pixels without discrimination. A majority of these misclassified pixels are not topology-critical. They are near the boundary of the foreground. Please see

Fig. 6.11 for an illustration.

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

Figure 6.11: Illustration of comparison between the proposed DMT-loss and a simple reweighted cross entropy loss. Please refer to Fig. 6.8(b) for the likelihood map of a baseline method without topological guarantee. **(a)** an input neuron image. **(b)** The topologically critical structures from the likelihood, captured by the proposed discrete Morse algorithm. These structures will be used in the DMT-Loss. **(c)** ground truth. **(d)** The FP/FN pixels identified by simple re-weighting cross entropy loss.

Table 6.1: Quantitative results for different models on several 2D datasets.

Method	Accuracy	DICE	ARI	VOI	Betti Error
ISBI13					
DIVE	0.9642 ± 0.0018	0.9658 ± 0.0020	0.6923 ± 0.0134	2.790 ± 0.025	3.875 ± 0.326
UNet	0.9631 ± 0.0024	0.9649 ± 0.0057	0.7031 ± 0.0256	2.583 ± 0.078	3.463 ± 0.435
Mosin.	0.9578 ± 0.0029	0.9623 ± 0.0047	0.7483 ± 0.0367	1.534 ± 0.063	2.952 ± 0.379
TopoLoss	0.9569 ± 0.0031	**0.9689 ± 0.0026**	0.8064 ± 0.0112	1.436 ± 0.008	**1.253 ± 0.172**
DMT	0.9625 ± 0.0027	**0.9712 ± 0.0047**	**0.8289 ± 0.0189**	**1.176 ± 0.052**	**1.102 ± 0.203**
CREMI					
DIVE	0.9498 ± 0.0029	0.9542 ± 0.0037	0.6532 ± 0.0247	2.513 ± 0.047	4.378 ± 0.152
UNet	0.9468 ± 0.0048	0.9523 ± 0.0049	0.6723 ± 0.0312	2.346 ± 0.105	3.016 ± 0.253
Mosin.	0.9467 ± 0.0058	0.9489 ± 0.0053	0.7853 ± 0.0281	1.623 ± 0.083	1.973 ± 0.310
TopoLoss	0.9456 ± 0.0053	0.9596 ± 0.0029	0.8083 ± 0.0104	1.462 ± 0.028	**1.113 ± 0.224**
DMT	0.9475 ± 0.0031	**0.9653 ± 0.0019**	**0.8203 ± 0.0147**	**1.089 ± 0.061**	**0.982 ± 0.179**
ISBI12					
DIVE	0.9640 ± 0.0042	0.9709 ± 0.0029	0.9434 ± 0.0087	1.235 ± 0.025	3.187 ± 0.307
UNet	0.9678 ± 0.0021	0.9699 ± 0.0048	0.9338 ± 0.0072	1.367 ± 0.031	2.785 ± 0.269
Mosin.	0.9532 ± 0.0063	0.9716 ± 0.0022	0.9312 ± 0.0052	0.983 ± 0.035	1.238 ± 0.251
TopoLoss	0.9626 ± 0.0038	0.9755 ± 0.0041	0.9444 ± 0.0076	0.782 ± 0.019	**0.429 ± 0.104**
DMT	0.9593 ± 0.0035	**0.9796 ± 0.0033**	**0.9527 ± 0.0052**	**0.671 ± 0.027**	**0.391 ± 0.114**
DRIVE					
DIVE	0.9549 ± 0.0023	0.7543 ± 0.0008	0.8407 ± 0.0257	1.936 ± 0.127	3.276 ± 0.642
UNet	0.9452 ± 0.0058	0.7491 ± 0.0027	0.8343 ± 0.0413	1.975 ± 0.046	3.643 ± 0.536
Mosin.	0.9543 ± 0.0047	0.7218 ± 0.0013	0.8870 ± 0.0386	1.167 ± 0.026	2.784 ± 0.293
TopoLoss	0.9521 ± 0.0042	0.7621 ± 0.0036	0.9024 ± 0.0113	1.083 ± 0.006	**1.076 ± 0.265**
DMT	0.9495 ± 0.0036	**0.7733 ± 0.0039**	**0.9077 ± 0.0021**	**0.876 ± 0.038**	**0.873 ± 0.402**
CrackTree					
DIVE	0.9854 ± 0.0052	0.6530 ± 0.0017	0.8634 ± 0.0376	1.570 ± 0.078	1.576 ± 0.287
UNet	0.9821 ± 0.0097	0.6491 ± 0.0029	0.8749 ± 0.0421	1.625 ± 0.104	1.785 ± 0.303
Mosin.	0.9833 ± 0.0067	0.6527 ± 0.0010	0.8897 ± 0.0201	1.113 ± 0.057	1.045 ± 0.214
TopoLoss	0.9826 ± 0.0084	0.6732 ± 0.0041	**0.9291 ± 0.0123**	0.997 ± 0.011	**0.672 ± 0.176**
DMT	0.9842 ± 0.0041	**0.6811 ± 0.0047**	**0.9307 ± 0.0172**	**0.901 ± 0.081**	**0.518 ± 0.189**
Road					
DIVE	0.9734 ± 0.0077	0.6743 ± 0.0051	0.8201 ± 0.0128	2.368 ± 0.203	3.598 ± 0.783
UNet	0.9786 ± 0.0052	0.6612 ± 0.0016	0.8189 ± 0.0097	2.249 ± 0.175	3.439 ± 0.621
Mosin.	0.9754 ± 0.0043	0.6673 ± 0.0044	0.8456 ± 0.0174	1.457 ± 0.096	2.781 ± 0.237
TopoLoss	0.9728 ± 0.0063	0.6903 ± 0.0038	0.8671 ± 0.0068	1.234 ± 0.037	**1.275 ± 0.192**
DMT	0.9744 ± 0.0049	**0.7056 ± 0.0022**	**0.8819 ± 0.0104**	**1.092 ± 0.129**	**0.995 ± 0.301**

Table 6.2: Comparison with same backbones.

	UNet	DIVE
TopoLoss	1.451± 0.216	1.113 ± 0.224
DMT-Loss	**0.982 ± 0.179**	**0.956 ± 0.142**

Table 6.3: Quantitative results for different models on several 3D datasets.

Method	Accuracy	DICE	ARI	VOI	Betti Error
ISBI13					
3D DIVE	0.9723 ± 0.0021	0.9681 ± 0.0043	0.8719 ± 0.0189	1.208 ± 0.149	2.375 ± 0.419
3D UNet	0.9746 ± 0.0025	0.9701 ± 0.0012	$\mathbf{0.8956 \pm 0.0391}$	1.123 ± 0.091	1.954 ± 0.585
MALA	0.9701 ± 0.0018	0.9699 ± 0.0013	$\mathbf{0.8945 \pm 0.0481}$	0.901 ± 0.106	1.103 ± 0.207
3D TopoLoss	0.9689 ± 0.0031	0.9752 ± 0.0045	$\mathbf{0.9043 \pm 0.0283}$	0.792 ± 0.086	0.972 ± 0.245
DMT	0.9701 ± 0.0026	$\mathbf{0.9803 \pm 0.0019}$	$\mathbf{0.9149 \pm 0.0217}$	$\mathbf{0.634 \pm 0.086}$	$\mathbf{0.812 \pm 0.134}$
CREMI					
3D DIVE	0.9503 ± 0.0061	0.9641 ± 0.0011	0.8514 ± 0.0387	1.219 ± 0.103	2.674 ± 0.473
3D UNet	0.9547 ± 0.0038	0.9618 ± 0.0026	0.8322 ± 0.0315	1.416 ± 0.097	2.313 ± 0.501
MALA	0.9472 ± 0.0027	0.9583 ± 0.0023	0.8713 ± 0.0286	1.109 ± 0.093	1.114 ± 0.309
3D TopoLoss	0.9523 ± 0.0043	0.9672 ± 0.0010	0.8726 ± 0.0194	1.044 ± 0.128	1.076 ± 0.206
DMT	0.9529 ± 0.0031	$\mathbf{0.9731 \pm 0.0045}$	$\mathbf{0.9013 \pm 0.0202}$	$\mathbf{0.891 \pm 0.099}$	$\mathbf{0.726 \pm 0.187}$
3Dircadb					
3D DIVE	0.9618 ± 0.0054	0.6097 ± 0.0034	/	/	4.571 ± 0.505
3D UNet	0.9632 ± 0.0009	0.5898 ± 0.0025	/	/	4.131 ± 0.483
MALA	0.9546 ± 0.0033	0.5719 ± 0.0043	/	/	2.982 ± 0.105
3D TopoLoss	0.9561 ± 0.0019	0.6138 ± 0.0029	/	/	2.245 ± 0.255
DMT	0.9587 ± 0.0023	$\mathbf{0.6257 \pm 0.0021}$	/	/	$\mathbf{1.415 \pm 0.305}$

Table 6.4: Comparison with other simpler choices.

Method	Accuracy	Betti Error
DMT	0.9475	0.982
Canny edge detection	0.9386	2.971
Ridge detection	0.9443	2.507

Table 6.5: Results with different noise levels.

Method	Accuracy	Betti Error
DMT	0.9475	0.982
Gaussian $(10\%, \delta)$	0.9393	1.086
Gaussian $(20\%, \delta)$	0.9272	1.391

Chapter 7

Learning Probabilistic Topological Representations Using Discrete Morse Theory

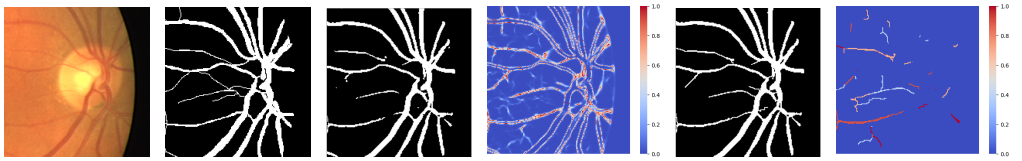
In the previous chapters, we have developed algorithms that can improve topology-aware segmentation accuracy, while all these methods are essentially learning pixel-wise representations. In this chapter, we propose to learn topological/structural representations directly.

7.1 Introduction

Accurate segmentation of fine-scale structures, e.g., vessels, neurons, and membranes is crucial for downstream analysis. In recent years, topology-inspired losses have been proposed to improve structural accuracy [2, 8, 39, 94, 109]. These losses identify topologically critical locations at which a segmentation network is error-prone, and force the network to improve its prediction at these critical locations.

However, these loss-based methods are still not ideal. They are based on a standard segmentation network, and thus *only learn pixel-wise feature representations*. This causes several issues. First, a standard segmentation network makes pixel-wise predictions. Thus, at the inference stage, topological errors, e.g. broken connections, can still happen, even though they may be mitigated by the topology-inspired losses. Another issue is in uncertainty estimation, i.e., estimating how certain a segmentation network is at different locations. Uncertainty maps can

direct the focus of human annotators for efficient proofreading [205,206]. However, for fine-scale structures, the existing pixel-wise uncertainty map is not effective. As shown in Fig. 7.1(d), every pixel adjacent to a vessel branch is highly uncertain, in spite of whether the branch is salient or not. What is more desirable is a structural uncertainty map that can highlight uncertain branches (e.g., Fig. 7.1(f)).



(a) Image (b) GT (c) Pro.UNet (d) Pixel Uncer. (e) Ours (f) Stru. Uncer.

Figure 7.1: Illustration of structural segmentation and structure-level uncertainty. Compared with Probabilistic-UNet [7] (Fig. 7.1(c)-(d)), the proposed method is able to generate a structure-preserving segmentation map (Fig. 7.1(e)), and structure-level uncertainty (Fig. 7.1(f)).

To fundamentally address these issues, we propose to directly model and reason about the structures. In this chapter, we propose *the first deep learning based method that directly learns the topological/structural* [1] *representation of images*. To move from pixel space to structural space [2], we apply the classic discrete Morse theory [16, 17] to decompose an image into a Morse complex, consisting of structural elements like branches, patches, etc. These structural elements are hypothetical structures one can infer from the input image. Their combinations constitute a space of structures arising from the input image. See Fig. 7.2(c) for an illustration.

For further reasoning with structures, we propose to learn a probabilistic model over the structural space. The challenge is that the space consists of exponentially many branches and is thus of very high dimension. To reduce the learning burden, we introduce the theory of persistent homology [118,120,200] for structure pruning. Each branch has its own persistence measuring its relative saliency. By continuously thresholding the complete Morse complex in terms of persistence, we obtain a sequence of Morse complexes parameterized by the persistence threshold, ϵ. See

[1] We will be using the terms topology/topological and structure/structural interchangeably in this chapter.

[2] Structures/branches instead of pixels construct the space and the operations are conducted at the structure/branches level instead of pixel level.

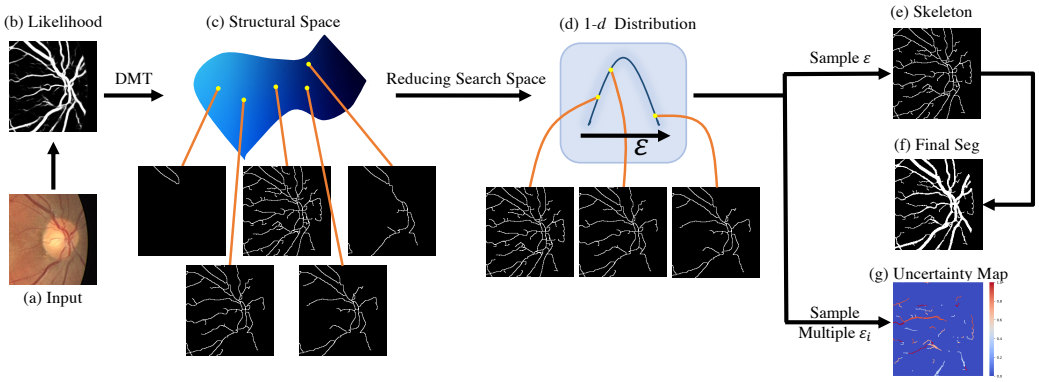


Figure 7.2: The probabilistic topological/structural representation. **(a)** is a sample input, **(b)** is the predicted likelihood map from the deep neural network, **(c)** is the whole structural space obtained by running a discrete Morse theory algorithm on the likelihood map, **(d)** the 1-d structural family parametrized by the persistence threshold ϵ, as well as a Gaussian distribution over ϵ, **(e)** a sampled skeleton, **(f)** the final structural segmentation map generated using the skeleton sample, and **(g)** the uncertainty map generated by multiple segmentations.

Fig. 7.2(d). By learning a Gaussian over ϵ, we learn a parametric probabilistic model over these structures.

This parametric probabilistic model over structural space allows us to make direct structural predictions via sampling (Fig. 7.2(e)), and to estimate the empirical structure-level uncertainty via sampling (Fig. 7.2(g)). The benefit is two-fold: First, direct prediction of structures will ensure the model outputs always have structural integrity, even at the inference stage. This is illustrated in Fig. 7.1(e). Samples from the probabilistic model are all feasible structural hypotheses based on the input image, with certain variations at uncertain locations. This is in contrast to state-of-the-art methods using pixel-wise representations (Fig. 7.1(c)-(d)). Note the original output structure (Fig. 7.2(e), also called skeleton) is only 1-pixel wide and may not serve as a good segmentation output. In the inference stage, we use a post-processing step to grow the structures without changing topology as the final segmentation prediction (Fig. 7.2(f)). More details are provided in Sec. 7.2.2 and Fig. 7.5.

Second, the probabilistic structural model can be seamlessly incorporated into semi-automatic interactive annotation/proofreading workflows to facilitate large scale annotation of these complex structures. This is especially important in the biomedical domain where fine-scale structures are notoriously difficult to annotate,

due to the complex 2D/3D morphology and low contrast near extremely thin structures. Our probabilistic model makes it possible to identify uncertain structures for efficient interactive annotation/proofreading. Note that the structural space is crucial for uncertainty reasoning. As shown in Fig. 7.1(f) and Fig. 7.2(g), our structural uncertainty map highlights uncertain branches for efficient proofreading. On the contrary, the traditional pixel-wise uncertainty map (Fig. 7.1(d)) is not helpful at all; it highlights all pixels on the boundary of a branch.

The main contributions of this chapter are:

1. We propose the first deep learning method which learns structural representations, based on discrete Morse theory and persistent homology.
2. We learn a probabilistic model over the structural space, which facilitates different tasks such as topology-aware segmentation, uncertainty estimation and interactive proofreading.
3. We validate our method on various datasets with *rich and complex structures*. It outperforms state-of-the-art methods in both deterministic and probabilistic categories.

7.2 Method

Our key innovation is to restructure the output of a neural network so that it is indeed making predictions over a space of structures. This is achieved through insights into the topological structures of an image and the usage of several important tools in topological data analysis.

To move from pixel space to structural space, we apply discrete Morse theory to decompose an image into a Morse complex, consisting of structures like branches, patches, etc. For simplification, we will use the term "branch" to denote a single piece of Morse structure. These Morse branches are the hypothetical structures one can infer from the input image. This decomposition is based on a likelihood function produced by a pixel-wise segmentation network trained in parallel. Thus it is of good quality, i.e., the structures are close enough to the true structures.

Any binary labeling of these Morse branches is a legitimate segmentation; we call it a *structural segmentation*. But for full-scope reasoning of the structural space, instead of classifying these branches one-by-one, we would like to have the full inference, i.e., predicting a probability distribution for each branch. To further reduce the degrees of freedom to make the inference easier, we apply persistent homology to filter these branches with regard to their saliency. This gives us a linear size family of structural segmentations, parameterized by a threshold ϵ.

Finally, we learn a 1D Gaussian distribution for the ϵ as our probabilistic model. This gives us the opportunity not only to sample segmentations, but also to provide a probability for each branch, which can be useful in downstream tasks including proofreading. In Sec. 7.2.1, we introduce the discrete Morse theory and how to construct the space of Morse structures. We also explain how to use persistent homology to reduce the search space of reasoning into a 1-parameter family. In Sec. 7.2.2, we will provide details on how our deep neural network is constructed to learn the probabilistic model over the structural space, as illustrated in Fig. 7.4.

7.2.1 Constructing the Structural Space

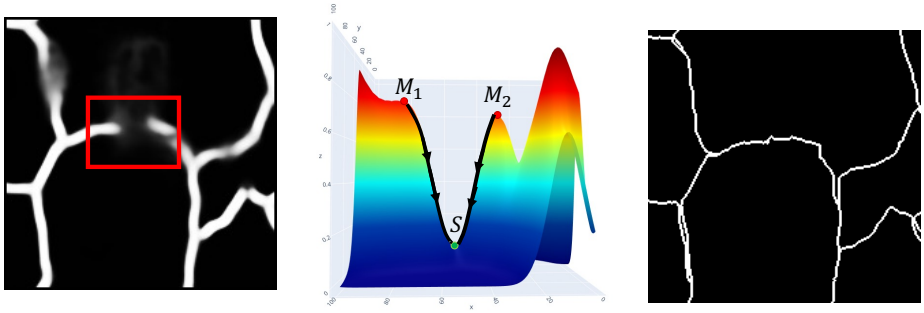
In this section, we explain how to construct a structural representation space using discrete Morse theory. The resulting structural representation space will be used to build a probabilistic model. We will then discuss how to reduce the structural space into a 1-parameter family of structural segmentations, using persistent homology. We assume a 2D input image, although the method naturally extends to 3D images.

Given a reasonably clean input (e.g., the likelihood map of a deep neural network, Fig. 7.2(b)), we treat the 2D likelihood as a terrain function, and Morse theory [16] can help to capture the structures regardless of weak/blurred conditions. See Fig. 7.3 for an illustration. The weak part of a line in the continuous map can be viewed as the local dip in the mountain ridge of the terrain. In the language of Morse theory, the lowest point of this dip is a saddle point (S in Fig. 7.3(b)), and the mountain ridges which are connected to the saddle point (M_1S and M_2S) are called the stable manifolds of the saddle point.

We mainly focus on 2D images in this chapter, although extending to 3D images is natural. We consider two dimensional continuous functions $f : \mathbb{R}^2 \to \mathbb{R}$. For a point $x \in \mathbb{R}^2$, the gradient can be computed as $\nabla f(x) = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}]^T$. We call a point $x = (x_1, x_2)$ *critical* if $\nabla f(x) = 0$. For a Morse function defined on $\mathbb{R}^2$, a critical point could be a minimum, a saddle or a maximum.

Consider a continuous line (the red rectangle region in Fig. 7.3(a)) in a 2D likelihood map. Imagine if we put a ball on one point of the line, then $-\nabla f(x)$ indicates the direction in which the ball will flow down. By definition, the ball will eventually flow to the critical points where $\nabla f(x) = 0$. The collection of points whose ball eventually flows to p ($\nabla f(p) = 0$) is defined as the stable manifold (denoted as $S(p)$) of point p. Intuitively, for a 2D function f, the stable manifold $S(p)$ of a minimum p is the entire valley of p (similar to the watershed algorithm); similarly, the stable manifold $S(q)$ of a saddle point q consists of the whole ridge

line which connects two local maxima and goes through the saddle point. See Fig. 7.3(b) as an illustration.



(a) Likelihood map (b) Terrain view (c) Morse complex

Figure 7.3: **(a)** shows a sample likelihood map from the deep neural network, and **(b)** is the terrain view of the red patch in **(a)** and illustrates the stable manifold of a saddle point in 2D case for a line-like structure. **(c)** is the 2D Morse complex generated by DMT from **(a)**.

For a link-like structure, the stable manifold of a saddle contains the topological structures (usually curvilinear) of the continuous likelihood map predicted by deep neural networks, and they are exactly what we want to recover from noisy images. In practice, we adopt the discrete version of Morse theory for images.

Discrete Morse Theory. Take a 2D image as a 2-dimensional cubical complex. A 2-dimensional cubical complex then contains 0-, 1-, and 2-dimensional cells, which correspond to vertices (pixels), edges, and squares, respectively. In the setting of discrete Morse theory (DMT) [17, 199], a pair of adjacent cells, termed discrete gradient vectors, compose the gradient vector. Critical points ($\nabla f(x) = 0$) are those critical cells that are not in any discrete gradient vectors. In the 2D domain, a minimum, a saddle, and a maximum correspond to a critical vertex, a critical edge, and a critical square respectively. A 1-stable manifold (the stable manifold of a saddle point) in 2D corresponds to a *V-path*, i.e., connecting two local maxima and a saddle. See Fig. 7.3(b). And the Morse complex generated by the DMT algorithm is illustrated in Fig. 7.3(c).

Constructing the Full Structural Space. In this way, by using discrete Morse theory, for a likelihood map from the deep neural network, we can extract all the stable manifolds of the saddles, whose compositions constitute the full structural space. *Formally, we call any combinations of these stable manifolds a structure*. Fig. 7.2(c) illustrates 5 different structures. This structural space, however, is of

exponential size. Assume there are N pieces of stable manifolds/branches for a given likelihood map. Any combinations of these stable manifolds/branches will be a potential structure. We will have 2^N possible structures in total. This can be computationally prohibitive to construct and model. We need a principled way to reduce the structural search space.

Reducing the Structural Search Space with Persistent Homology. We propose to use persistent homology [118, 120, 200] to reduce the structural search space. Persistent homology is an important tool for topological data analysis [19, 20]. Intuitively, we grow a Morse complex by gradually including more and more discrete elements (called cells) from empty. A branch of the Morse complex is a special type of cell. Other types include vertices, patches, etc. Cells will be continuously added to the complex. New branches will be born and existing branches will die. The persistence algorithm [20] pairs up all these critical cells as birth and death pairs. The difference of their function values is essentially the life time of the specific topological structure/branch, which is called the *persistence*. The importance of a branch is associated with its persistence. Intuitively, the longer the persistence of a specific branch is, the more important the branch is.

Recall that our original construction of the structural space considers all possible combinations of branches, and thus can have exponentially many combinations. Instead, we propose to only select branches with high persistence as important ones. By doing this, we will be able to prune the less important/noisy branches very efficiently and recover the branches with true signals. Specifically, the structure pruning is done via *Morse cancellation* operation. The persistence thresholding provides us the opportunity to obtain a *structural space of linear size*. We start with the complete Morse complex, and continuously increase the threshold ϵ. At each threshold, we obtain a structure by filtering with ϵ and only keeping the branches whose persistence is above ϵ. This gives a sequence of structures parametrized by ϵ. As shown in Fig. 7.2(d), the family of structures represents different structural densities.

The one-parameter space allows us to easily learn a probabilistic model and carry out various inference tasks such as segmentation, sampling, uncertainty estimation, and interactive proofreading. Specifically, we will learn a Gaussian distribution over the persistence threshold ϵ, $\epsilon \sim N(\mu, \sigma)$. Denote the persistence of a branch b as ϵ_b. Any branch b belongs to the structure map M (we also call the structure map M a structural segmentation) as long as its persistence is smaller or equal to the maximal persistence threshold of M, i.e., $b \in M$ if and only if $\epsilon_b \leq \epsilon_M$, where ϵ_M is used to generate M ($\epsilon_M \geq \max_{b \in M} \epsilon_b$). More details will be provided in Sec. 7.2.2.

Morse Cancellation. As the predicted likelihood map is noisy, the extracted discrete gradient field $\mathsf{M}(K)$ could also be noisy. Fortunately, the discrete Morse theory provides an elegant way to cancel critical simplicial pairs and ignore the unimportant Morse branches. Particularly, if there is a unique V-path $\pi = \delta = \delta_0, \gamma_1, \delta_1, \ldots, \delta_s, \gamma_{s+1} = \gamma$ from δ to γ, then the pair of critical simplices $\langle \delta^{(p+1)}, \gamma^p \rangle$ is *cancellable* . By removing all V-pairs along these path, and adding (δ_{i-1}, γ_i) to $\mathsf{M}(K)$ for any $i \in [1, s+1]$, the *Morse cancellation operation* reverses all V-pairs along this path. In this way, neither δ nor γ is critical after the cancellation operation and we can prune/remove the corresponding stable manifold/branch. More details can be found in [8].

Approximation of Morse Structures for Volume Data. In the 2D setting, the stable manifold of saddles is composed of curvilinear structures, and the captured Morse structures will essentially contain the *non-boundary edges*, which fits well with vessel data. However, the output structures should always be *boundary edges* for volume data, which can't be dealt with the original discrete Morse theory. Consequently, we approximate the Morse structures of 2D volume data with the boundaries of the stable manifolds of local minima. As mentioned above, the stable manifold of a local minimum p in the 2D setting corresponds to the whole valley, and the boundaries of these valleys construct the approximation of the Morse structures for volume data. Similar to the original discrete Morse theory, we also introduce a persistence threshold parameter ϵ and use persistent homology to prune the less important branches. The details of the proposed persistent-homology filtered topology watershed algorithm are illustrated in Alg. 1.

7.2.2 Neural Network Architecture

In this section, we introduce our neural network that learns a probabilistic model over the structural representation to obtain structural segmentations. See Fig. 7.4 for an illustration of the overall pipeline.

Since the structural reasoning needs a sufficiently clean input to construct discrete Morse complexes, our method first obtains such a likelihood map by training a segmentation branch which is supervised by the standard segmentation loss, cross-entropy loss. Formally, $L_{seg} = L_{bce}(Y, S(X; \omega_{seg}))$, in which $S(X; \omega_{seg})$ is the output likelihood map, ω_{seg} is the segmentation branch's weight.

The output likelihood map, $S(X; \omega_{seg})$, is used as the input for the discrete Morse theory algorithm (DMT), which generates a discrete Morse complex consisting of all possible Morse branches from the likelihood map. Thresholding these branches using persistent homology with different ϵ's will produce

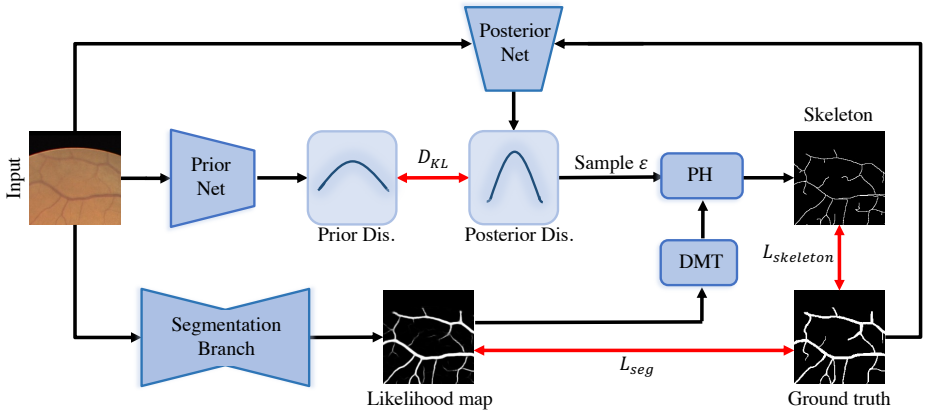


Figure 7.4: The overall workflow of the training stage. The red arrows indicate supervision.

different structures. We refer to the DMT computation and the persistent homology thresholding operation as f_{DMT} and f_{PH}. So given a likelihood map $S(X;\omega_{seg})$ and a threshold ϵ, we can generate a structure (which we call a skeleton): $S_{skeleton}(\epsilon) = f_{PH}(f_{DMT}(S(X;\omega_{seg}));\epsilon)$. Next, we discuss how to learn the probabilistic model. Recall we want to learn a Gaussian distribution over the persistent homology threshold, $\epsilon \sim N(\mu,\sigma)$. The parameters μ and σ are learned by a neural network called the *posterior network*. The network uses the input image X and the corresponding ground truth mask Y as input, and outputs the parameters $\mu(X,Y;\omega_{post})$ and $\sigma(X,Y;\omega_{post})$. ω_{post} is the parameter of the network.

During training, at each iteration, we draw a sample ϵ from the distribution ($\epsilon \sim N(\mu,\sigma)$). Using the sample ϵ, together with the likelihood map, we can generate the corresponding sampled structure, $S_{skeleton}(\epsilon)$. This skeleton will be compared with the ground truth for supervision. To compare a sampled skeleton, $S_{skeleton}(\epsilon)$, with ground truth Y, we use the skeleton to mask both Y and the likelihood map $S(X;\omega_{seg})$, and then compare the skeleton-masked ground truth and the likelihood using cross-entropy loss: $L_{bce}(Y \circ S_{skeleton}(\epsilon), S(X;\omega_{seg}) \circ S_{skeleton}(\epsilon))$.

To learn the distribution, we use the expected loss:

$$L_{skeleton} = \mathbb{E}_{\epsilon\sim N(\mu,\sigma)} L_{bce}(Y \circ S_{skeleton}(\epsilon), S(X;\omega_{seg}) \circ S_{skeleton}(\epsilon)) \quad (7.1)$$

The skeleton is actually the inference structure from the probabilistic model and the supervision of the skeleton loss ensures the topological correctness of the inference results. The loss can be backpropagated through the posterior network

through the reparameterization technique [207]. Note that this loss will also provide supervision to the segmentation network through the likelihood map.

Reparameterization Technique. We adopt the reparameterization technique of VAE to make the network differentiable and be able to backpropagate.

The posterior net randomly draws samples from the posterior distribution $\epsilon \sim N(\mu_{post}, \sigma_{post})$. To implement the posterior net as a neural network, we will need to backpropagate through random sampling. The issue is that backpropagation cannot flow through random nodes; to overcome this obstacle, we adopt the reparameterization technique proposed in [207].

Assuming the posterior is normally distributed, we can approximate it with another normal distribution. We approximate ϵ with normally distribution Z ($Z \sim N(0, \mathbf{I})$).

$$\epsilon \sim N(\mu, \sigma), \qquad \epsilon = \mu + \sigma Z. \tag{7.2}$$

Now instead of saying that ϵ is sampled from $Q(X, Y; \omega_{post})$, we can say ϵ is a function that takes parameter (Z,(μ, σ)) and these μ, σ come from deep neural network. Therefore all we need is partial derivatives w.r.t. μ, σ, and Z is irrelevant for taking derivatives for backpropagation.

Learning a Prior Network from the Posterior Network. Although our posterior network can learn the distribution well, it relies on the ground truth mask Y as input. This is not available at the inference stage. To address this issue, inspired by Probabilistic-UNet [7], we use another network to learn the distribution of ϵ with only the image X as input. We call this network the *prior net*. We denote by P the distribution using parameters predicted by the prior network, and denote by Q the distribution predicted by the posterior network.

During the training, we want to use the prior net to mimic the posterior net; and then in the inference stage, we can use the prior net to obtain a reliable distribution over ϵ with only the image X. Thus, we incorporate the Kullback-Leibler divergence of these two distributions, $D_{KL}(Q||P) = \mathbb{E}_{\epsilon \sim Q}(\log \frac{Q}{P})$, which measures the differences of prior distribution P ($N(\mu_{prior}, \sigma_{prior})$) and the posterior distribution Q ($N(\mu_{post}, \sigma_{post})$).

Training the Neural Network. The final loss is composed by the standard segmentation loss, the skeleton loss $L_{skeleton}$, and the KL divergence loss, with two hyperparameters α and β to balance the three terms,

$$L(X, Y) = L_{seg} + \alpha L_{skeleton} + \beta D_{KL}(Q||P) \tag{7.3}$$

The network is trained to jointly optimize the segmentation branch and the probabilistic branch (containing both prior and posterior nets) simultaneously. During the

training stage, the KL divergence loss (D_{KL}) pushes the prior distribution towards the posterior distribution. The training scheme is also illustrated in Fig. 7.4.

Inference Stage: Generating Structure-Preserving Segmentation Maps. In the inference stage, given an input image, we are able to produce an unlimited number of plausible structure-preserving skeletons via sampling. We use a post-processing step to grow the 1-pixel wide structures/skeletons without changing their topology as the final structural segmentation. Specifically, the skeletons are overlaid on the binarized initial segmentation map (Fig. 7.5(c)), and only the connected components which exist in the skeletons are kept as the final segmentation maps (Fig. 7.5(e)). In this way, each plausible skeleton (Fig. 7.5(d)) generates one final segmentation map (Fig. 7.5(e)) and it has exactly the same topology as the corresponding skeleton. The pipeline of the procedure is illustrated in Fig. 7.5.

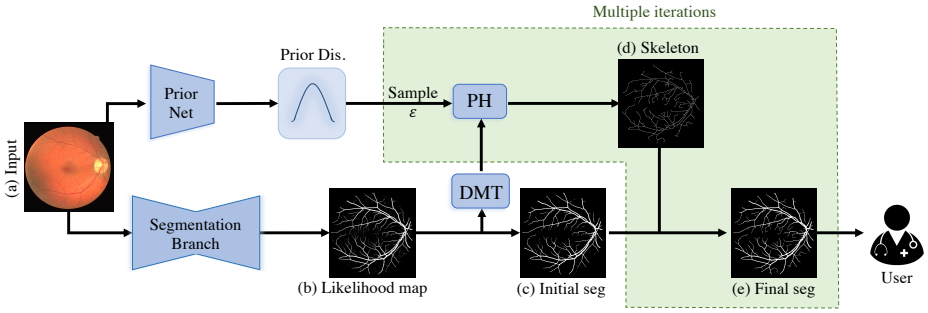


Figure 7.5: The inference and interactive annotation/proofreading pipeline.

Uncertainty of Structures. Given a learned prior distribution, P, over the family of structural segmentations, we can naturally calculate the probability of each Morse structure/branch. Recall a branch b has its persistence ϵ_b. And the prior probability of a structural segmentation map M is $P(\epsilon_M)$, in which ϵ_M is used to generate M. Also, any branch b whose persistence is smaller or equal to the maximal persistence threshold of M belongs to M, i.e., $b \in M$ if and only if $\epsilon_b \leq \epsilon_M$. Thus we have $\epsilon_M \geq \max_{b \in M} \epsilon_b$. Therefore, the probability of a branch b being in a segmentation map M such as $\epsilon_M \sim P$ follows a Bernoulli distribution with the probability $Pr(b)$ being the cumulative distribution function (CDF) of P, $CDF_P(\epsilon_b) = P(\epsilon \leq \epsilon_b)$. This can be directly calculated at inference, and the absolute difference of the CDF from 0.5 is the confidence (which equals 1-uncertainty) of the Morse structure/branch b.

7.3 Experiments

Our method directly makes predictions and inferences on structures rather than on pixels. This can significantly benefit downstream tasks. While probabilities of structures can certainly be used for further analysis of the structural system, in this chapter we focus on both automatic image segmentation and semi-automatic annotation/proofreading tasks. On automatic image segmentation, we show that direct prediction can ensure topological integrity even better than previous topology-aware losses. This is not surprising as our prediction is on structures. On the semi-automatic proofreading task, we show our structure-level uncertainty can assist human annotators to obtain satisfying segmentation annotations in a much more efficient manner than previous methods.

7.3.1 Automatic Topology-Aware Image Segmentation

We introduce the datasets, evaluation metrics, and baselines used in this chapter to demonstrate the effectiveness of the proposed method.

7.3.2 Datasets

We use three datasets to validate the efficacy of the proposed method: **ISBI13** [169] (volume), **CREMI** (volume), and **DRIVE** [172] (vessel). Both volume and vessel datasets are used to validate the efficacy of the proposed method, and the details of the datasets have been introduced in Sec. 3.3.1. We use a 3-fold cross-validation for all the methods to report the numbers over the validation set.

7.3.3 Evaluation Metrics

We use four different evaluation metrics: **Dice score**, **ARI**, **VOI**, and **Betti number error**. More details about the evaluation metrics have been provided in Sec. 3.3.2 and Sec. 5.2.7.

7.3.4 Baselines

We compare the proposed method with two kinds of baselines: 1) Standard segmentation baselines: **DIVE** [1], **UNet** [6], **UNet-VGG** [94], **TopoLoss** [2] and **DMT** [8]. 2) Probabilistic-based segmentation methods: **Dropout UNet** [156] and

Probabilistic-UNet [7]. The details of Probabilistic-based segmentation methods are listed as follows:

- Dropout UNet [156] dropouts the three inner-most encoder and decoder blocks with a probability of 0.5 during both the training and inference.
- Probabilistic-UNet [7] introduces a probabilistic segmentation method by combining UNet with a VAE.

For all methods, we generate binary segmentations by thresholding predicted likelihood maps at 0.5.

Quantitative and Qualitative Results. Tab. 7.1 shows the quantitative results compared to several baselines. Note that for deterministic methods, the numbers are computed directly based on the outputs; while for probabilistic methods, we generate five segmentation masks and report the averaged numbers over the five segmentation masks for each image (for both the baselines and the proposed method). We use the t-test (95% confidence interval) to determine the statistical significance and highlight the significantly better results. From the table, we can observe that the proposed method achieves significantly better performances in terms of topology-aware metrics (ARI, VOI, and Betti Error).

Fig. 7.6 shows qualitative results. Compared with DMT [8], our method is able to produce a set of true structure-preserving segmentation maps, as illustrated in Fig. 7.6(e-g). Note that compared with the existing topology-aware segmentation methods, our method is more capable of recovering the weak connections by using Morse skeletons as hints.

Ablation Study of Loss Weights. We observe that the performances of our method are quite robust to the loss weights α and β. As the learned distribution over the persistence threshold might affect the final performances, we conduct an ablation study in terms of the weight of KL divergence loss (β) on the DRIVE dataset. The results are reported in Fig. 7.7. When $\beta = 10$, the model achieves slightly better performance in terms of VOI (0.804 ± 0.047, the smaller the better) than other choices. Note that, for all the experiments, we set $\alpha = 1$.

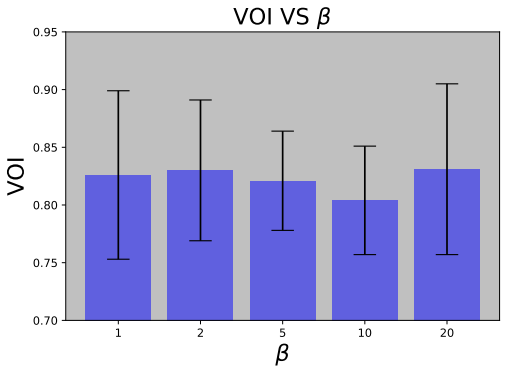


Figure 7.7: Ablation study for β.

Illustration of the Structure-level Uncertainty. We also explore the structure-level uncertainty with the proposed method here. We show three sampled masks (Fig. 7.8(c-e)) in the inference stage for a given image (Fig. 7.8(a)), and the structure-level uncertainty map (Fig. 7.8(f)). Note that in practice, for simplification, different from Sec. 7.2.2, we empirically generate an uncertainty map by taking variance across all the samples (the number is 10 in our case). Different from pixel-wise uncertainty, each small branch has the same uncertainty value as our method. By looking at the original image, we find that the uncertainties are usually caused by the weak signals of the original image. The weak signals of the original image make the model difficult to predict these locations correctly and confidently, especially structure wise. Actually, this also makes sense in real cases. Different from natural images, even experts can not always reach a consensus for biomedical image annotation [208, 209].

Fig. 7.1 shows the comparison of traditional pixel-wise uncertainty and the proposed structure-level uncertainty. Specifically, Fig. 7.1(c) is a sampled segmentation result by Prob.-UNet [7], and Fig. 7.1(d) is the pixel uncertainty map from Prob.-UNet [7]. Different from traditional pixel-wise uncertainty, our proposed structure-level uncertainty (Fig. 7.1(f)) can focus on the structures.

We also overlay the structure-level uncertainty (Fig. 7.1(f)) on the original image (Fig. 7.1(a)), which is shown in Fig. 7.9. By comparison with the original image, we can observe that the structure-level uncertainty is mainly caused by the weak signals in the original image.

The Advantage of the Joint Training and Optimization. Another straightforward alternative to the proposed approach is to use the discrete Morse theory to post-process the continuous likelihood map obtained from the standard segmentation networks.

In this way, we can still obtain structure-preserving segmentation maps, but there are two main issues: 1) if the segmentation network itself is structure-agnostic, we'll not be able to generate satisfactory results even with the postprocessing, and 2) we have to manually choose the persistence threshold to prune the unnecessary branches for each image, which is cumbersome and unrealistic in practice. The proposed joint training strategy overcomes both of these issues. First, during the training, we incorporate the structure-aware loss ($L_{skeleton}$). Consequently, the trained segmentation branch itself is structure-aware essentially. On the other hand, with the prior and posterior nets, we are able to learn a reliable distribution of the persistence threshold (ϵ) given an image in the inference stage. Sampling over the distribution makes it possible to generate satisfactory structure-preserving segmentation maps within a few trials (the inference won't take long), which is

more much efficient.

We conduct an empirical experiment to demonstrate the advantage of joint training and optimization. For the postprocessing, given the predicted likelihood maps from the standard segmentation networks, we randomly choose five persistence thresholds and generate the segmentation masks separately, and select the most reasonable one as the final segmentation mask to report the performances. The results in Tab. 7.2 demonstrate the advantage of our joint training and optimization strategy.

7.3.5 Semi-Automatic Efficient Annotation/Proofreading with User Interaction

Proofreading is a struggling while essential step for the annotation of images with fine-scale structures. We develop a semi-automatic interactive annotation or proofreading pipeline based on the proposed method. As the proposed method is able to generate structural segmentations (Fig. 7.8(c)-(e)), the annotators can efficiently proofread one image with rich structures by removing the unnecessary/redrawing the missing structures with the help of structure-level uncertainty map (Fig. 7.8(f)). The whole inference and structure-aware interactive image annotation pipeline is illustrated in Fig. 7.5.

We conduct empirical experiments to demonstrate the efficiency of proofreading by using the proposed method. We randomly select a few samples from the ISBI13 dataset and simulate the user interaction process. For both the proposed and baseline methods, we get started from the final segmentations and correct *one misclassified branch* each time. For the deterministic method (DMT), the user draws one false-negative or erases one false-positive branch for each click. For the pixel-wise probabilistic method (Prob.-UNet), the user does the same while taking the uncertainty map as guidance. For the proposed method, the user checks each branch based on the descending order of structure-level uncertainty. VOI is used to evaluate the performances.

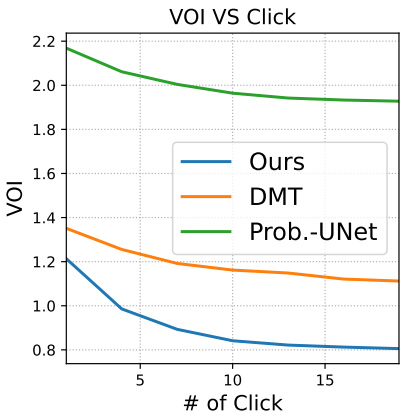


Figure 7.10: Interaction simulation.

Fig. 7.10 shows the comparative results of semi-automatic proofreading with user interactions. By always checking branches with the highest uncertainty, the proposed method clearly achieves better results and improves the results much faster than baseline methods. Our developed pipeline achieves higher efficiency because of the following two perspectives: 1) the generated structural segmentations are essentially topology/structure-preserving; 2) the proposed method provides the structure-level uncertainty map to guide the human proofreading.

7.4 Applications: Mitochondria Annotation

Mitochondria play a vital role as the main energy suppliers for cellular functions, making them indispensable for metabolism. Accurate measurement of the size and shape of mitochondria is not only critical for fundamental neuroscience investigations [210] but also valuable for clinical studies examining various diseases [211, 212].

Electron microscopy (EM) provides detailed snapshots of cellular and subcellular ultrastructure at high resolutions. Recent advancements in volume EM have made it possible to obtain imaging of larger specimens [213, 214], and deep learning algorithms play an active role in enabling quantitative analyses [55, 123]. These methods have also been applied to investigate organellar structures in various systems [215, 216], leading to unprecedented biological insights on a large scale. The typical deep learning workflow for such applications involves densely annotating within regions of interest (2D or 3D), training a model based on the annotations, conducting inference on the unannotated dataset, and then involving humans to proofread the model's output [123, 217]. Although achieving impressive performances, both predicting and proofreading are conducted at the pixel level, which is not efficient enough.

The original method is developed for curvilinear structures. In this section, we mainly focus on the problem of human-in-the-loop mitochondria annotation. In order to extend from curvilinear structure data to non-curvilinear structure data, we propose to adopt distance transform and transform the instance segmentation task to a boundary segmentation problem. By measuring the existing confidence/uncertainty of the boundaries, we are able to measure the probability of instances getting connected or not. Based on the proposed probabilistic method, we aim to develop a semi-automatic human-in-the-loop mitochondria annotation pipeline. Our intuition is, together with the mitochondria segmentation mask, the generated probability map can be used to guide the proofreading process and

develop the semi-automatic human-in-the-loop annotation pipeline.

7.4.1 Distance Transform for Mitochondria

As aforementioned, we propose to transform the discretely labeled masks to distance maps for mitochondria data. Given the ground-truth label Y of the mitochondria, let S_V be the set of pixels/voxels on the mitochondria surface, which can be defined by

$$S_V = \{v | y_v = 1, \exists u \in \mathcal{N}(v), y_u = 0\}, \tag{7.4}$$

where $\mathcal{N}(v)$ denotes the 4 or 6-neigobor voxels (in 2D or 3D scenarios) of v. The distance transform map is formally defined as:

$$d_v = \begin{cases} \min\limits_{u \in S_V} ||v - u|| & \text{if } y_v = 0, \\ 0 & \text{if } y_v = 1. \end{cases} \tag{7.5}$$

For each voxel v, the distance transform operation assigns it a distance transform value which is the nearest distance from v to the mitochondria surface S_V.

To learn more accurate distance transform maps, we additionally impose a distance loss term used to train the network, which indicates a penalty if the predicted distance transform value is different from its counterpart of ground-truth. Details will be introduced later.

7.4.2 Segmentation Branch

Since structural reasoning needs a curvilinear-like structure to construct discrete Morse complexes, the proposed method first obtains such a likelihood map by training a segmentation branch. More specifically, we adopt distance transform to transform the binary mask to continuous distance maps. Besides enforcing the penalty on the predicted likelihood map, we also impose an additional penalty term on the distance maps. Formally,

$$L_{seg} = L_{bce}(Y, S(X; \omega_{seg})) + \lambda L_{bce}(d^{GT}, d^{pred}), \tag{7.6}$$

in which Y is the ground truth. X is the input, ω_{seg} denotes the parameters of the segmentation branch, and $S(X; \omega_{seg})$ is the output likelihood map. d_{GT} and d_{pred} denote the distance maps for binary ground truth and binarized predicted result, respectively. See Fig. 7.11 for an illustration.

In Fig. 7.11(d), we also show the medial axis of the background region. The medial axis partitions the whole image/patch into different sub-regions, and each sub-region corresponds to one mitochondria component (illustrated in 2D). The distance map of the binarized predicted result, d_{pred}, is a curvilinear structure and thus can be fed to the discrete Morse theory algorithm (DMT).

The discrete Morse theory (DMT) algorithm can decompose the distance map into many more small Morse branches, and each branch denotes a possible boundary to separate different mitochondria components. Formally, we call any combination of these branches a sample from the structural space, and each branch has a specific persistence value. Similarly, a threshold value ϵ can be used to filter out the most salient branches, and different ϵ's result in different branch combinations, each of them corresponding to one specific mitochondria segmentation result.

7.4.3 Skeleton Extraction

The DMT computation is referred to as d_{DMT}^{pred}, and the persistent homology thresholding is referred to as d_{PH}^{pred}. So given a distance map d_{PH}^{pred} and a threshold ϵ, a structure/skeleton can be generated as follow:

$$S_{skeleton}(\epsilon) = d_{PH}^{pred}(d_{DMT}^{pred}; \epsilon). \tag{7.7}$$

Besides the segmentation branch, we also want to make sure the $S_{skeleton}(\epsilon)$ is correctly predicted. More specifically, we impose additional supervision regarding the predicted skeleton. We use the skeleton $S_{skeleton}(\epsilon)$ to mask both the predicted distance map d^{pred} and ground truth distance map d^{GT}. And then we compute the binary cross-entropy loss over the sampled skeleton

$$L_{bce}(d^{GT} \circ S_{skeleton}(\epsilon), S(d^{pred} \circ S_{skeleton}(\epsilon)), \tag{7.8}$$

in which $\circ$ denotes the Hadamard product. One of our goals is to learn a 1-D Gaussian distribution, $\epsilon \sim N(\mu, \sigma)$, regarding the persistent homology threshold. Involving the 1-D Gaussian distribution, more formally, we have the expected loss:

$$\begin{aligned} L_{skeleton} = \mathbb{E}_{\epsilon \sim N(\mu,\sigma)} L_{bce}(d^{GT} \circ S_{skeleton}(\epsilon), \\ S(d^{pred} \circ S_{skeleton}(\epsilon)). \end{aligned} \tag{7.9}$$

This loss term also supervised the training of the segmentation branch via the predicted distance map. The backpropagation can be achieved through the reparameterization technique in [207].

We also employ the VAE framework to learn better distributions over ϵ. Thus, the Kullback-Leibler divergence of these two distributions is calculated,

$$D_{KL}(Q||P) = \mathbb{E}_{\epsilon\sim Q}(\log \frac{Q}{P}), \tag{7.10}$$

which measures how distance between the prior distribution P ($N(\mu_{prior}, \sigma_{prior})$) and the posterior distribution Q ($N(\mu_{post}, \sigma_{post})$).

7.4.4 Training Neural Network

Our final loss consists of standard segmentation loss (including the distance map term), the skeleton term $L_{skeleton}$, and the KL divergence term.

$$L(X, Y) = L_{seg} + \alpha L_{skeleton} + \beta D_{KL}(Q||P), \tag{7.11}$$

and it can be rewritten as,

$$\begin{aligned} L(X, Y) = L_{bce}(Y, S(X; \omega_{seg})) + \lambda L_{bce}(d^{GT}, d^{pred}) + \\ \alpha L_{skeleton} + \beta D_{KL}(Q||P), \end{aligned} \tag{7.12}$$

The parameters λ, α, and β are weights to balance the four different loss terms.

7.4.5 Experiment Design and Results

Datasets: We conduct our experiments on popular mitochondria datasets, MitoEM dataset [218]. **MitoEM [218]**: MitoEM dataset consists of two volumes with resolutions of $8 \times 8 \times 30\ nm^3$, which come from human (MitoEM-H) and rat (MitoEM-R) cortices, respectively. The authors cropped a (30 μm)^3 sub-volume to contain the mitochondria. The carefully curated dataset contains complex mitochondria without introducing much of the domain adaptation problem due to the diverse image appearance. Each subset has a training set ($400 \times 4096 \times 4096$) and a validation set ($100 \times 4096 \times 4096$).
Baselines: We compare the proposed method with two kinds of baselines: 1) Standard segmentation baselines: **DIVE** [1], **UNet** [6], and 2) Probabilistic-based segmentation methods: **Dropout UNet** [156], **Probabilistic-UNet** [7]. The details of these baselines have been introduced in Sec. 7.3.4.
Evaluation Metrics: We use four different evaluation metrics: **Jaccard-index coefficient**, **Dice score**, **AP-75**, and **VOI**. The details of **Dice score** and **VOI** can be found in Sec. 3.3.2 and Sec. 5.2.7. The other two metrics are as followings:

- *Jaccard-index coefficient*: Jaccard-index coefficient score is usually used to gauge the similarity and diversity of sample sets, and it's often used for segmentation tasks.
- *AP-75*: Following [218], we also choose AP-75, which requires at least 75% intersection over union (IoU) with the ground truth for a detection to be a true positive.

Results: Table 7.3 shows the quantitative results regarding the segmentation, comparing to baselines. Note that for deterministic methods (standard segmentation methods), the numbers are computed directly; while for probabilistic-base segmentation methods, five segmentation masks are generated at once, and the averaged numbers over the five segmentation masks for each image (for both the baselines and the proposed method). The proposed method obviously outperforms the baseline methods.

Fig. 7.13 shows qualitative results. Our method is able to predict both segmentation maps (Fig. 7.13(d)) and distance maps (Fig. 7.13(e)) accurately.

7.4.6 Human-in-the-loop Structure-Aware Mitochondria Annotation Pipeline

Based on segmentation results and the boundary uncertainty estimation, we are able to develop an efficient human-in-the-loop structure-aware mitochondria annotation pipeline. The rationale is that, by always focusing on the most uncertain branches, the human annotations will be able to proofread the segmentation results efficiently.

We conduct empirical experiments on the MitoEm-H dataset to demonstrate the efficiency of the proposed mitochondria annotation pipeline. For both the proposed and baseline methods, we get started from the final segmentations and conduct one click each time. For the pixel-wise probabilistic method (Dropout UNet and Prob.-UNet), the user focuses on one branch each time. For the proposed method, the

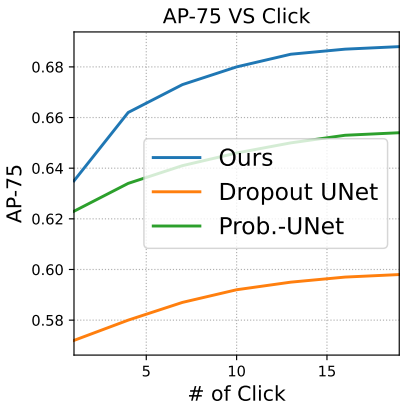


Figure 7.14: Human-in-the-loop annotation illustration. Experiments are conducted on the MitoEM-H dataset.

user checks branch by branch based on the descending order of branch uncertainty. AP-75 is used to evaluate the performances.

Fig. 7.14 shows the results of the human-in-the-loop structure-aware mitochondria annotation pipeline. By always checking the most uncertain branches, the proposed method obviously outperforms baseline methods, which is not surprising, as the developed pipeline is always focusing on the whole branches instead of pixels.

7.5 Conclusion

Instead of making pixel-wise predictions, we propose to learn structural representation with a probabilistic model to obtain structural segmentations. Specifically, we construct the structure space by leveraging classical discrete Morse theory. We then build a probabilistic model to learn a distribution over structures. In the inference stage, we are able to generate a set of structural segmentations and explore the structure-level uncertainty, which is beneficial for interactive proofreading. We further extend the proposed method from curvilinear structure data to non-curvilinear structure data (mitochondria). Extensive experiments have been conducted to demonstrate the efficacy of the proposed method.

Algorithm 1: Persistent-Homology filtered Topology Watershed Algorithm

Input: a grid 2D image, and a threshold θ

Output: Morse structures for volume data

Definition: $G = (V, E)$ denote a graph; $f(v)$ is the intensity value of node v; lower_star(v) = $\{(u, v) \in E | f(u) < f(v)\}$; $cc(v)$ is the connected component id of node v.

PD =$\emptyset$; Build the proximity graph (4-connectivity) for 2D grid image;
$U = V$ sorted according to $f(v)$; T a sub-graph, which includes all the nodes and edges whose value $< t$.
for v in U **do**
  $t = f(v)$, $T = T + \{v\}$
  for (u, v) in lower_star(v) **do**
    Assert $u \in T$
    if $cc(u) = cc(v)$ **then**
      Edge_tag(u, v) = loop edge
      Continue
    else
      Edge_tag(u, v) = tree edge
      younger_cc = $\arg\max_{w=cc(u),cc(v)} f(w)$
      older_cc = $\arg\min_{w=cc(u),cc(v)} f(w)$
      pers = t-*f(younger_cc)*
      if pers >= θ **then**
        Edge_tag(u, v) = *watershed edge*
        Continue
      end if
      for w in younger_cc **do**
        $cc(w)$ = older_cc
      end for
      PD = PD + (f(younger_cc), t)
    end if
  end for
end for
return Membrane_vertex_set = $\cup$ vertices of watershed_edge_set

Table 7.1: Quantitative results for different models on three different biomedical datasets.

Method	Dice ↑	ARI ↑	VOI ↓	Betti Error ↓
ISBI13 (Volume)				
DIVE	0.9658 ± 0.0020	0.6923 ± 0.0134	2.790 ± 0.025	3.875 ± 0.326
UNet	0.9649 ± 0.0057	0.7031 ± 0.0256	2.583 ± 0.078	3.463 ± 0.435
UNet-VGG	0.9623 ± 0.0047	0.7483 ± 0.0367	1.534 ± 0.063	2.952 ± 0.379
TopoLoss	0.9689 ± 0.0026	0.8064 ± 0.0112	1.436 ± 0.008	1.253 ± 0.172
DMT	**0.9712 ± 0.0047**	0.8289 ± 0.0189	1.176 ± 0.052	1.102 ± 0.203
Dropout UNet	0.9591 ± 0.0031	0.7127 ± 0.0181	2.483 ± 0.046	3.189 ± 0.371
Prob.-UNet	0.9618 ± 0.0019	0.7091 ± 0.0201	2.319 ± 0.041	3.019 ± 0.233
Ours	0.9637 ± 0.0032	**0.8417 ± 0.0114**	**1.013 ± 0.081**	**0.972 ± 0.141**
CREMI (Volume)				
DIVE	0.9542 ± 0.0037	0.6532 ± 0.0247	2.513 ± 0.047	4.378 ± 0.152
UNet	0.9523 ± 0.0049	0.6723 ± 0.0312	2.346 ± 0.105	3.016 ± 0.253
UNet-VGG	0.9489 ± 0.0053	0.7853 ± 0.0281	1.623 ± 0.083	1.973 ± 0.310
TopoLoss	0.9596 ± 0.0029	0.8083 ± 0.0104	1.462 ± 0.028	1.113 ± 0.224
DMT	0.9653 ± 0.0019	0.8203 ± 0.0147	1.089 ± 0.061	0.982 ± 0.179
Dropout UNet	0.9518 ± 0.0018	0.6814 ± 0.0202	2.195 ± 0.087	3.190 ± 0.198
Prob.-UNet	0.9531 ± 0.0022	0.6961 ± 0.0115	1.901 ± 0.107	2.931 ± 0.177
Ours	**0.9681 ± 0.0016**	**0.8475 ± 0.0043**	**0.935 ± 0.069**	**0.919 ± 0.059**
DRIVE (Vessel)				
DIVE	0.7543 ± 0.0008	0.8407 ± 0.0257	1.936 ± 0.127	3.276 ± 0.642
UNet	0.7491 ± 0.0027	0.8343 ± 0.0413	1.975 ± 0.046	3.643 ± 0.536
UNet-VGG	0.7218 ± 0.0013	0.8870 ± 0.0386	1.167 ± 0.026	2.784 ± 0.293
TopoLoss	0.7621 ± 0.0036	0.9024 ± 0.0113	1.083 ± 0.006	1.076 ± 0.265
DMT	0.7733 ± 0.0039	0.9077 ± 0.0021	0.876 ± 0.038	0.873 ± 0.402
Dropout UNet	0.7410 ± 0.0019	0.8331 ± 0.0152	2.013 ± 0.072	3.121 ± 0.334
Prob.-UNet	0.7429 ± 0.0020	0.8401 ± 0.1881	1.873 ± 0.081	3.080 ± 0.206
Ours	**0.7814 ± 0.0026**	**0.9109 ± 0.0019**	**0.804 ± 0.047**	**0.767 ± 0.098**

Table 7.2: Quantitative results for comparison of postprocessing on DRIVE dataset.

Method	Dice ↑	ARI ↑	VOI ↓	Betti Error ↓
Postprocessing	0.7653 ± 0.0052	0.8841 ± 0.0046	1.165 ± 0.086	1.249 ± 0.388
Ours	**0.7814 ± 0.0026**	**0.9109 ± 0.0019**	**0.804 ± 0.047**	**0.767 ± 0.098**

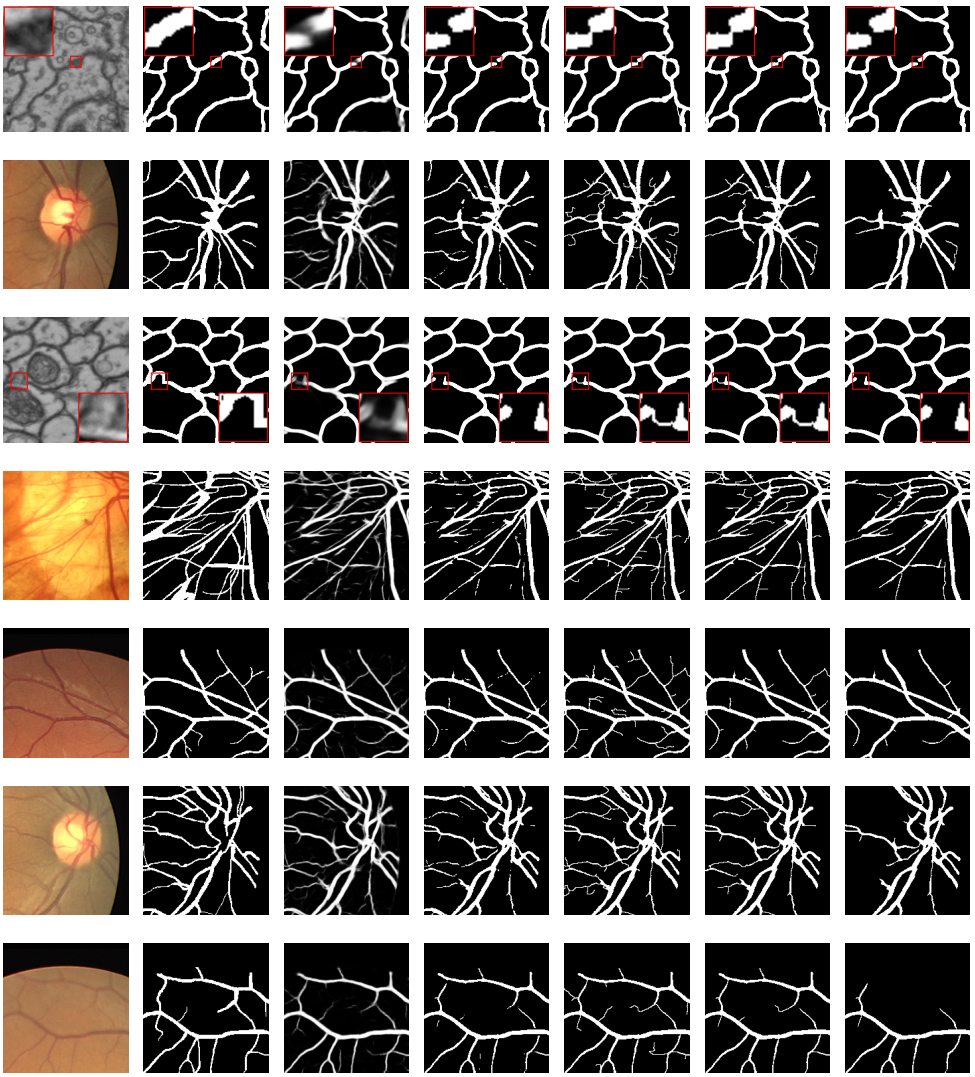
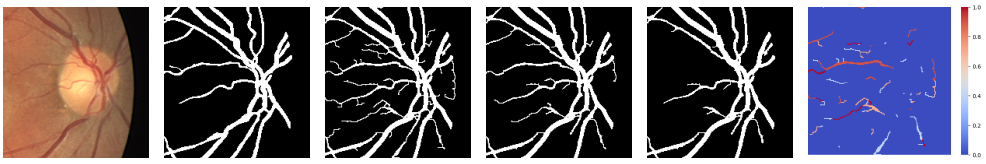
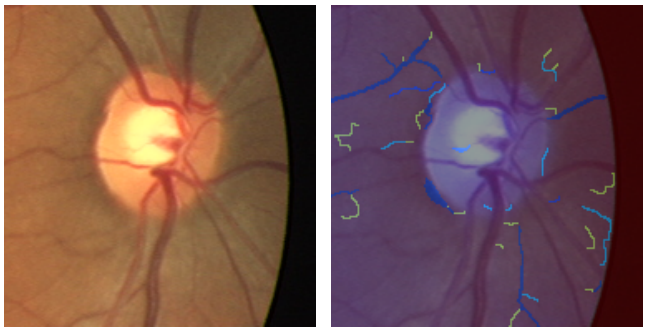


Figure 7.6: Qualitative results of our method compared to DMT-loss [8]. From left to right: **(a)** image, **(b)** ground truth, **(c)** continuous likelihood map and **(d)** thresholded binary mask for DMT [8], and **(e-g)** three sampled segmentation maps generated by our method.



(a) Image (b) GT (c) Sample1 (d) Sample2 (e) Sample3 (f) Uncertainty

Figure 7.8: An illustration of structure-level uncertainty.



(a) Image (b) Overlay

Figure 7.9: Overlaying the structure-level uncertainty on the original image.

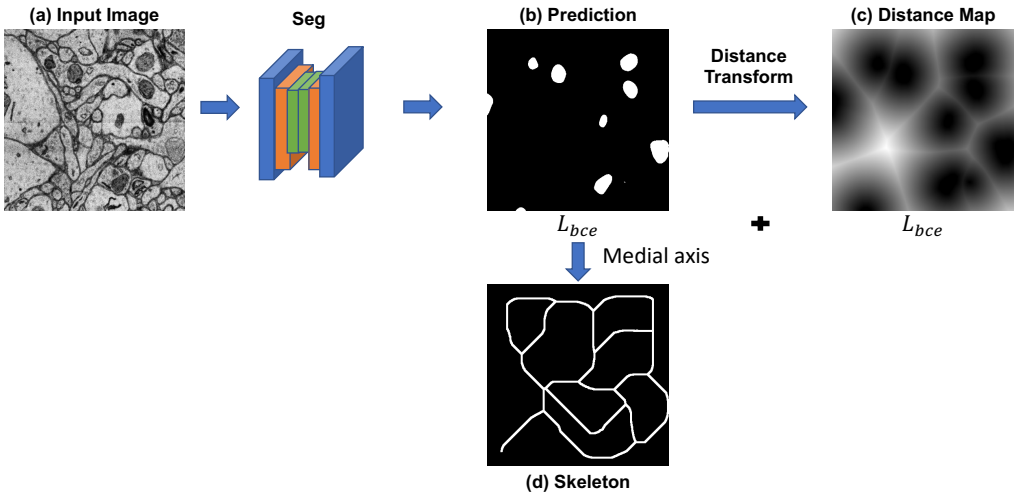


Figure 7.11: Illustration of the segmentation branch in 2D view.

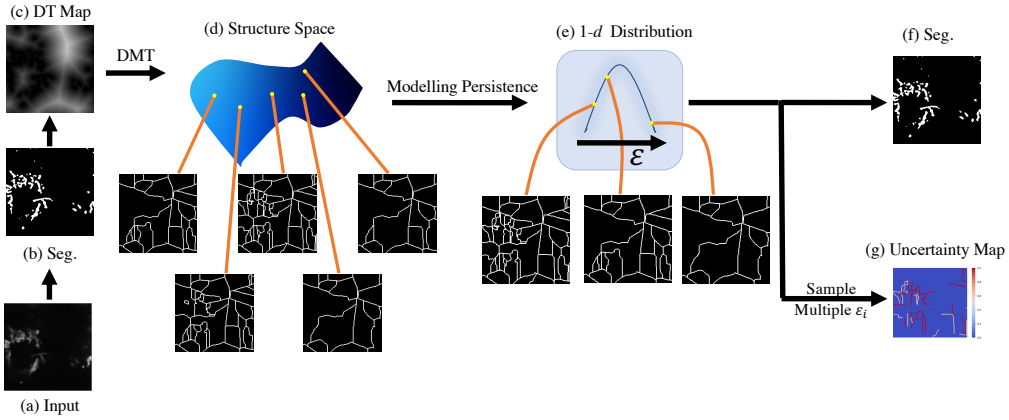


Figure 7.12: The overall framework of the proposed method. Part of the framework credits to [9].

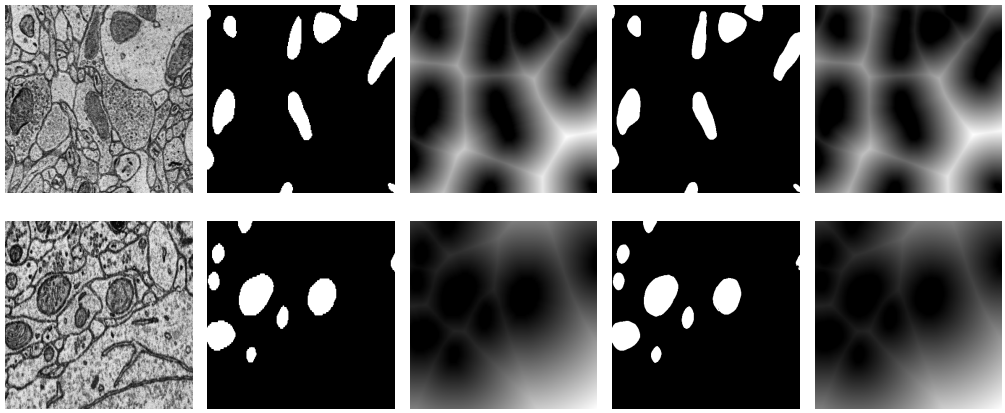


Figure 7.13: Segmentation results of the proposed method. From left to right: **(a)** image, **(b)** ground truth, **(c)** d^{GT}, **(d)** segmentation maps generated by our method, and **(e)** d^{pred}.

Table 7.3: Quantitative results for different models on mitochondria datasets.

Method	Jaccard ↑	Dice ↑	AP-75 ↑	VOI ↓
MitoEM-R				
UNet	0.821	0.897	0.481	0.262
DIVE	0.792	0.874	0.407	0.291
Dropout UNet	0.833	0.905	0.496	0.238
Prob.-UNet	0.841	0.911	0.541	0.284
Ours	**0.847**	**0.912**	**0.604**	**0.173**
MitoEM-H				
UNet	0.816	0.884	0.583	0.344
DIVE	0.788	0.850	0.482	0.318
Dropout UNet	0.820	0.891	0.572	0.378
Prob.-UNet	0.825	0.892	0.623	0.298
Ours	**0.839**	**0.905**	**0.635**	**0.216**

Chapter 8

Conclusion and Future Work

In the previous chapters, we have introduced our research on *Learning Topological Representations for Deep Image Segmentation*, in five chapters:

- *Topology-Preserving Deep Image Segmentation*, Chapter 3 [2].
- *Trigger Hunting with a Topological Prior for Trojan Detection*, Chapter 4 [219].
- *Structure-Aware Image Segmentation with Homotopy Warping*, Chapter 5 [220].
- *Topology-Aware Segmentation Using Discrete Morse Theory*, Chapter 6 [8].
- *Learning Probabilistic Topological Representations Using Discrete Morse Theory*, Chapter 7 [9].

My research has been focused on understanding images from the topology/structure view. By using persistent homology, we have been proposing differentiable topological loss to segment with correct topology [2, 188]. By enforcing topological constraints into the trojan detection context [219], we further explore the power of the topological loss to recover triggers with as few connected components as possible. To avoid the critical points which are not really relevant to topological errors, we further introduce another warping strategy to efficiently identify critical points [220]. Instead of identifying a sparse set of critical points at every epoch, we use discrete Morse theory to identify critical structures instead of critical points [8]. Furthermore, we propose to learn topological/structural representations directly [9].

Throughout my thesis research on learning topological representations on deep image understanding, I believe topology plays an important role in analyzing

images, especially biomedical images, with rich structures. Exploring the geometry/topology/structure of various data will be helpful to achieve my ultimate goal — building AI systems that can efficiently assist in disease diagnosis and treatment. Though making solid progress, there is still a long way to go before I reach my long-term goal. Below, we discuss a few directions I plan to pursue in the next stage.

- ***Efficiency of Using Data***: By using topological tools [2, 8, 188, 219–223], I have explored the data efficiently by taking their topological properties into consideration. However, as the field of topology-driven image analysis continues to evolve, I am deeply motivated to explore even deeper into the realm of data efficiency. More specifically, I plan to develop novel methods to optimize the utilization of data, capitalizing on its intrinsic topological/geometric/structural information to enhance the efficiency of image analysis methodologies. By using data-driven techniques, I strive to create novel algorithms with both concrete theoretical foundations and strong empirical results for imaging data under different contexts, especially biomedical scenarios.

- ***Uncertainty Estimation and Its Applications***: Besides achieving good performances, we usually want to measure how the model is confident of its predictions, which is important in different scenarios. *A good model should both know what it knows and what it does not know*. Building on my recent work regarding structure-wise uncertainty estimation for curvilinear structure data [9, 224] and confidence estimation using unlabeled data [225, 226], I want to further expand the boundary of uncertainty estimation and its wide-ranging applications. As deep learning methods have been generalized in different contexts, understanding and quantifying the uncertainty of deep models has emerged as a crucial aspect. By demystifying the uncertainty of models, I aim to not only enhance the reliability of predictive models but also extend their applications to diverse domains.

- ***Deep Learning Based Quantification Analysis***: Based on my work on topology-preserving deep image segmentation and topology-aware uncertainty estimation in Chapter 6 and Chapter 7, I am ready for quantifying how the accurate topology/geometry aware segmentation results affect the downstream analysis. By extracting topology/geometry-informed features, we can do a number of interesting analyses, such as diagnosing retinal diseases, predicting aortic aneurysm eruption risk, and so on. Fig. 8.1 is an illustration

pipeline for the downstream analysis. This is an ongoing direction and will be explored further.

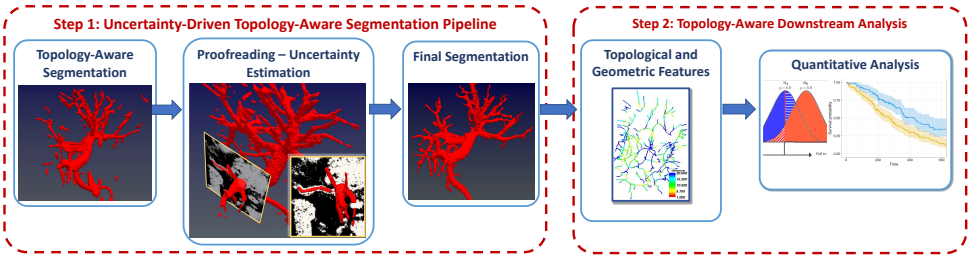


Figure 8.1: Workflow for uncertainty-driven topology-aware segmentation, and the downstream topology/geometry aware analysis.

- ***Learning with Imperfect Data***: In my thesis, I have been focused on fully supervised scenarios with ground truth. While in practice especially in biomedical contexts, the gathering of labeled data can be cost-prohibitive and time-consuming, which usually requires domain knowledge [227]. The dependency on diverse, high-quality training datasets substantially limits model applicability to complex scenes where data from are imperfect including *missing modality*, and/or *human labeling limited*. Furthermore, the uncertainty estimation [225] I have been working on might provide hints for scenarios with imperfect data. Driven by this, I would like to answer the following question: 1) are we able to achieve satisfactory performances even under scenarios with imperfect data? 2) are we able to focus on the most unconfident samples/regions to facilitate the training with imperfect data? [227]

As aforementioned, over the next few years, I will continue to create novel algorithms with both concrete theoretical foundations and strong empirical results for imaging data under different contexts, especially biomedical scenarios, and build intelligent AI systems that can assist in diagnosis and disease treatment.

Publication List

(* indicates equal contribution)

1. Calibrating Uncertainty for Semi-Supervised Crowd Counting

   Chen Li, **Xiaoling Hu**, Shahira Abousamra, Chao Chen

   International Conference on Computer Vision (ICCV), 2023

2. Enhancing Modality-Agnostic Representations via Meta-Learning for Brain Tumor Segmentation

   Aishik Konwer, **Xiaoling Hu**, Xuan Xu, Joseph Bae, Chao Chen, Prateek Prasanna

   International Conference on Computer Vision (ICCV), 2023

3. Learning Probabilistic Topological Representations Using Discrete Morse Theory

   Xiaoling Hu, Dimitris Samaras, Chao Chen

   International Conference on Learning Representations (ICLR), 2023 (**Spotlight, notable-top-25%**)

4. Confidence Estimation Using Unlabeled Data

   Chen Li, **Xiaoling Hu**, Chao Chen

   International Conference on Learning Representations (ICLR), 2023

5. Structure-Aware Image Segmentation with Homotopy Warping

   Xiaoling Hu

   Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS), 2022

6. Learning Topological Interactions for Multi-Class Medical Image Segmentation

   Saumya Gupta*, **Xiaoling Hu***, James Kaan, Michael Jin, Mutshipay Mpoy, Katherine Chung, Gagandeep Singh, Mary Saltz, Tahsin Kurc, Joel Saltz, Apostolos Tassiopoulos, Prateek Prasanna, Chao Chen

   European Conference on Computer Vision (ECCV), 2022 (**Oral, 2.7%**)

7. Trigger Hunting with a Topological Prior for Trojan Detection

   Xiaoling Hu, Xiao Lin, Michael Cogswell, Yi Yao, Susmit Jha, Chao Chen

   International Conference on Learning Representations (ICLR), 2022

8. A Manifold View of Adversarial Risk

   Wenjia Zhang, Yikai Zhang, **Xiaoling Hu**, Mayank Goswami, Chao Chen, Dimitris Metaxas

   International Conference on Artificial Intelligence and Statistics (AISTATS), 2022

9. Topology-Attention ConvLSTM Network for 3D Image Segmentation

   Jiaqi Yang*, **Xiaoling Hu***, Chao Chen, Chialing Tsai

   International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2021

10. Topology-Aware Segmentation Using Discrete Morse Theory

    Xiaoling Hu, Yusu Wang, Li Fuxin, Dimitris Samaras, Chao Chen

    International Conference on Learning Representations (ICLR), 2021 (**Spotlight, 5.6%**)

11. 3D topology-preserving segmentation with Z-dimension multi-resolution representation

    Jiaqi Yang*, **Xiaoling Hu***, Chao Chen, Chialing Tsai

    IEEE International Symposium on Biomedical Imaging (ISBI), 2021

12. Topology-Preserving Deep Image Segmentation

    Xiaoling Hu, Li Fuxin, Dimitris Samaras, Chao Chen

    Thirty-third Conference on Neural Information Processing Systems (NeurIPS), 2019

References

[1] Ahmed Fakhry, Hanchuan Peng, and Shuiwang Ji. Deep models for brain em image segmentation: novel insights and improved performance. *Bioinformatics*, 2016.

[2] Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *Advances in neural information processing systems (NeurIPS)*, 2019.

[3] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2019.

[4] Yingqi Liu, Wen-Chuan Lee, Guanhong Tao, Shiqing Ma, Yousra Aafer, and Xiangyu Zhang. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[5] Wenbo Guo, Lun Wang, Xinyu Xing, Min Du, and Dawn Song. Tabor: A highly accurate approach to inspecting and restoring trojan backdoors in ai systems. In *International Conference on Data Mining (ICDM)*, 2019.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[7] Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus Maier-Hein, SM Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. A probabilistic u-net for segmentation of ambiguous images. In *Advances in neural information processing systems (NeurIPS)*, 2018.

[8] Xiaoling Hu, Yusu Wang, Li Fuxin, Dimitris Samaras, and Chao Chen. Topology-aware segmentation using discrete morse theory. In *International Conference on Learning Representations (ICLR)*, 2021.

[9] Xiaoling Hu, Dimitris Samaras, and Chao Chen. Learning probabilistic topological representations using discrete morse theory. In *International Conference on Learning Representations (ICLR)*, 2023.

[10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *International Conference on Computer Vision (ICCV)*, 2017.

[12] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[14] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[15] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning (ICML)*, 2016.

[16] John Milnor. *Morse theory.(AM-51)*. Princeton university press, 1963.

[17] Robin Forman. A user's guide to discrete morse theory. *Sém. Lothar. Combin*, 2002.

[18] Serguei Barannikov. The framed morse complex and its invariants. *American Mathematical Society Translations, Series 2*, 1994.

[19] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[20] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 2000.

[21] Frédéric Chazal, Vin De Silva, Marc Glisse, and Steve Oudot. The structure and stability of persistence modules. *arXiv preprint arXiv:1207.3674*, 2012.

[22] Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, Larry Wasserman, Sivaraman Balakrishnan, Aarti Singh, et al. Confidence sets for persistence diagrams. *The Annals of Statistics*, 2014.

[23] Gunnar Carlsson, Afra Zomorodian, Anne Collins, and Leonidas J Guibas. Persistence barcodes for shapes. *International Journal of Shape Modeling*, 2005.

[24] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. A general neural network architecture for persistence diagrams and graph classification. *arXiv preprint arXiv:1904.09378*, 2019.

[25] Azriel Rosenfeld. Digital topology. *The American Mathematical Monthly*, 1979.

[26] Pengxiang Wu, Chao Chen, Yusu Wang, Shaoting Zhang, Changhe Yuan, Zhen Qian, Dimitris Metaxas, and Leon Axel. Optimal topological cycles and their application in cardiac trabeculae restoration. In *Information Processing in Medical Imaging (IPMI)*, 2017.

[27] Roland Kwitt, Stefan Huber, Marc Niethammer, Weili Lin, and Ulrich Bauer. Statistical topological data analysis-a kernel perspective. In *Advances in neural information processing systems (NeurIPS)*, 2015.

[28] Eleanor Wong, Sourabh Palande, Bei Wang, Brandon Zielinski, Jeffrey Anderson, and P Thomas Fletcher. Kernel partial least squares regression for relating functional brain network topology to clinical measures of behavior. In *International Symposium on Biomedical Imaging (ISBI)*, 2016.

[29] Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba. Persistence-based clustering in riemannian manifolds. *Journal of the ACM (JACM)*, 2013.

[30] Xiuyan Ni, Novi Quadrianto, Yusu Wang, and Chao Chen. Composing tree graphical models with persistent homology features for clustering mixed-type data. In *International Conference on Machine Learning (ICML)*, 2017.

[31] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research (JMLR)*, 2017.

[32] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research (JMLR)*, 2015.

[33] Kush R Varshney and Karthikeyan Natesan Ramamurthy. Persistent topology of decision boundaries. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015.

[34] Qi Zhao, Ze Ye, Chao Chen, and Yusu Wang. Persistence enhanced graph neural network. In *Artificial Intelligence and Statistics Conference (AISTATS)*, 2020.

[35] Zuoyu Yan, Tengfei Ma, Liangcai Gao, Zhi Tang, and Chao Chen. Link prediction with persistent homology: An interactive view. In *International Conference on Machine Learning (ICML)*, 2021.

[36] Pengxiang Wu, Songzhu Zheng, Mayank Goswami, Dimitris Metaxas, and Chao Chen. A topological filter for learning with label noise. In *Advances in neural information processing systems (NeurIPS)*, 2020.

[37] Scott Kulp, Chao Chen, Dimitris Metaxas, and Leon Axel. Ventricular blood flow analysis using topological methods. In *International Symposium on Biomedical Imaging (ISBI)*, 2015.

[38] Frédéric Chazal, Vin De Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. Springer, 2016.

[39] James Clough, Nicholas Byrne, Ilkay Oksuz, Veronika A Zimmer, Julia A Schnabel, and Andrew King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[40] Shahira Abousamra, Minh Hoai, Dimitris Samaras, and Chao Chen. Localization in the crowd with topological constraints. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[41] Fan Wang, Huidong Liu, Dimitris Samaras, and Chao Chen. Topogan: A topology-aware generative adversarial network. In *European Conference on Computer Vision (ECCV)*, 2020.

[42] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Mandar Dixit. Connectivity-optimized representation learning via persistent homology. In *International Conference on Machine Learning (ICML)*, 2019.

[43] Christoph Hofer, Florian Graf, Bastian Rieck, Marc Niethammer, and Roland Kwitt. Graph filtration learning. In *International Conference on Machine Learning (ICML)*, 2020.

[44] Mathieu Carrière, Frédéric Chazal, Yuichi Ike, Théo Lacombe, Martin Royer, and Yuhei Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In *Artificial Intelligence and Statistics Conference (AISTATS)*, 2020.

[45] Chao Chen, Xiuyan Ni, Qinxun Bai, and Yusu Wang. A topological regularizer for classifiers via persistent homology. In *Artificial Intelligence and Statistics Conference (AISTATS)*, 2019.

[46] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2016.

[47] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *International Conference on Learning Representations (ICLR)*, 2017.

[48] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you

need. In *Advances in neural information processing systems (NeurIPS)*, 2017.

[50] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2019.

[51] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018.

[52] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *European Conference on Computer Vision (ECCV)*, 2018.

[53] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2015.

[54] Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems (NeurIPS)*, 2012.

[55] Jan Funke, Fabian Tschopp, William Grisaitis, Arlo Sheridan, Chandan Singh, Stephan Saalfeld, and Srinivas C Turaga. Large scale image segmentation with structured loss based deep learning for connectome reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[56] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020.

[57] Wei Shen, Bin Wang, Yuan Jiang, Yan Wang, and Alan Yuille. Multi-stage multi-recursive-input fully convolutional networks for neuronal boundary detection. In *International Conference on Computer Vision (ICCV)*, 2017.

[58] Bing Shuai, Zhen Zuo, Bing Wang, and Gang Wang. Scene segmentation with dag-recurrent neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[59] Piotr Dollar, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2006.

[60] Michael Maire, Pablo Arbelaez, Charless Fowlkes, and Jitendra Malik. Using contours to detect and localize junctions in natural images. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2008.

[61] Timothee Cour, Florence Benezit, and Jianbo Shi. Spectral segmentation with multiscale graph decomposition. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2005.

[62] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2012.

[63] Pedro HO Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *International Conference on Machine Learning (ICML)*, 2014.

[64] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *International Conference on 3D Vision (3DV)*, 2016.

[65] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017.

[66] Shuai Zhao, Yang Wang, Zheng Yang, and Deng Cai. Region mutual information loss for semantic segmentation. In *Advances in neural information processing systems (NeurIPS)*, 2019.

[67] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013.

[68] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 2021.

[69] Andrew Delong and Yuri Boykov. Globally optimal segmentation of multi-region objects. In *International Conference on Computer Vision (ICCV)*, 2009.

[70] Jörg Hendrik Kappes, Markus Speth, Gerhard Reinelt, and Christoph Schnörr. Higher-order segmentation via multicuts. *Computer Vision and Image Understanding (CVIU)*, 2016.

[71] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018.

[72] Jifeng Dai, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. Instance-sensitive fully convolutional networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[73] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017.

[74] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019.

[75] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018.

[76] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from edges to instances with multicut. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017.

[77] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2017.

[78] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NeurIPS)*, 2012.

[80] Jianxu Chen, Lin Yang, Yizhe Zhang, Mark Alber, and Danny Z Chen. Combining fully convolutional and recurrent neural networks for 3d biomedical image segmentation. In *Advances in neural information processing systems (NeurIPS)*, 2016.

[81] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[82] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2016.

[83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015.

[84] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

[85] Henghui Ding, Xudong Jiang, Ai Qun Liu, Nadia Magnenat Thalmann, and Gang Wang. Boundary-aware feature propagation for scene segmentation. In *International Conference on Computer Vision (ICCV)*, 2019.

[86] Hoel Kervadec, Jihene Bouchtiba, Christian Desrosiers, Eric Granger, Jose Dolz, and Ismail Ben Ayed. Boundary loss for highly unbalanced segmentation. In *Medical Imaging with Deep Learning (MIDL)*, 2019.

[87] Davood Karimi and Septimiu E Salcudean. Reducing the hausdorff distance in medical image segmentation with convolutional neural networks. *IEEE Transactions on Medical Imaging (TMI)*, 2020.

[88] Hei-Long Chan, Shi Yan, Lok-Ming Lui, and Xue-Cheng Tai. Topology-preserving image segmentation by beltrami representation of shapes. *Journal of Mathematical Imaging and Vision*, 2018.

[89] Yuan Xue, Hui Tang, Zhi Qiao, Guanzhong Gong, Yong Yin, Zhen Qian, Chao Huang, Wei Fan, and Xiaolei Huang. Shape-aware organ segmentation by predicting signed distance maps. *arXiv preprint arXiv:1912.03849*, 2019.

[90] Xu Chen, Bryan M Williams, Srinivasa R Vallabhaneni, Gabriela Czanner, Rachel Williams, and Yalin Zheng. Learning active contour models for medical image segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019.

[91] Aïcha BenTaieb and Ghassan Hamarneh. Topology aware fully convolutional networks for histology gland segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2016.

[92] Charan Reddy, Karthik Gopinath, and Herve Lombaert. Brain tumor segmentation using topological loss in convolutional networks. In *International Conference on Medical Imaging with Deep Learning–Extended Abstract Track*, 2019.

[93] Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1987.

[94] Agata Mosinska, Pablo Marquez-Neila, Mateusz Koziński, and Pascal Fua. Beyond the pixel-wise loss for topology-aware delineation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018.

[95] Xiao Han, Chenyang Xu, and Jerry L. Prince. A topology preserving level set method for geometric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2003.

[96] Carole Le Guyader and Luminita A Vese. Self-repelling snakes for topology-preserving segmentation models. *IEEE Transactions on Image Processing (TIP)*, 2008.

[97] Ganesh Sundaramoorthi and Anthony Yezzi. Global regularizing flows with topology preservation for active contours and polygons. *IEEE Transactions on Image Processing (TIP)*, 2007.

[98] Florent Ségonne. Active contours under topology control—genus preserving level sets. *International Journal of Computer Vision (IJCV)*, 2008.

[99] Mingchen Gao, Chao Chen, Shaoting Zhang, Zhen Qian, Dimitris Metaxas, and Leon Axel. Segmenting the papillary muscles and the trabeculae from high resolution cardiac ct through restoration of topological handles. In *Information Processing in Medical Imaging (IPMI)*, 2013.

[100] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Graph cut based image segmentation with connectivity priors. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2008.

[101] Sebastian Nowozin and Christoph H Lampert. Global connectivity potentials for random field models. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2009.

[102] Yun Zeng, Dimitris Samaras, Wei Chen, and Qunsheng Peng. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in n–d images. *Computer Vision and Image Understanding (CVIU)*, 2008.

[103] Chao Chen, Daniel Freedman, and Christoph H Lampert. Enforcing topological constraints in random field image segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2011.

[104] Bjoern Andres, Jörg H Kappes, Thorsten Beier, Ullrich Köthe, and Fred A Hamprecht. Probabilistic image segmentation with closedness constraints. In *International Conference on Computer Vision (ICCV)*, 2011.

[105] Jan Stuhmer, Peter Schroder, and Daniel Cremers. Tree shape priors with connectivity constraints using convex relaxation on general graphs. In *International Conference on Computer Vision (ICCV)*, 2013.

[106] Martin Ralf Oswald, Jan Stühmer, and Daniel Cremers. Generalized connectivity constraints for spatio-temporal 3d reconstruction. In *European Conference on Computer Vision (ECCV)*, 2014.

[107] Rolando Estrada, Carlo Tomasi, Scott C Schmidler, and Sina Farsiu. Tree topology estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014.

[108] Kevin Briggman, Winfried Denk, Sebastian Seung, Moritz N Helmstaedter, and Srinivas C Turaga. Maximin affinity learning of image segmentation. In *Advances in neural information processing systems (NeurIPS)*, 2009.

[109] Suprosanna Shit, Johannes C Paetzold, Anjany Sekuboyina, Ivan Ezhov, Alexander Unger, Andrey Zhylka, Josien PW Pluim, Ulrich Bauer, and Bjoern H Menze. cldice-a novel topology-preserving loss function for tubular structure segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2021.

[110] Jan Reininghaus, Stefan Huber, Ulrich Bauer, and Roland Kwitt. A stable multi-scale kernel for topological machine learning. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015.

[111] Genki Kusano, Yasuaki Hiraoka, and Kenji Fukumizu. Persistence weighted gaussian kernel for topological data analysis. In *International Conference on Machine Learning (ICML)*, 2016.

[112] Mathieu Carriere, Marco Cuturi, and Steve Oudot. Sliced wasserstein kernel for persistence diagrams. In *International Conference on Machine Learning (ICML)*, 2017.

[113] Chao Chen and Novi Quadrianto. Clustering high dimensional categorical data via topographical features. In *International Conference on Machine Learning (ICML)*, 2016.

[114] Chao Chen and Daniel Freedman. Topology noise removal for curve and surface evolution. In *International MICCAI Workshop on Medical Computer Vision*, 2010.

[115] Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J Guibas, and Steve Y Oudot. Proximity of persistence modules and their diagrams. In *Symposium on Computational Geometry (SoCG)*, 2009.

[116] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, 2018.

[117] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl. Deep learning with topological signatures. In *Advances in neural information processing systems (NeurIPS)*, 2017.

[118] Olaf Delgado-Friedrichs, Vanessa Robins, and Adrian Sheppard. Skeletonization and partitioning of digital images using discrete morse theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014.

[119] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and algorithms for constructing discrete morse complexes from grayscale digital images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2011.

[120] S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *ACM SIGSPATIAL*, 2015.

[121] T. Dey, J. Wang, and Y. Wang. Road network reconstruction from satellite images with machine learning supported by topological methods. In *ACM SIGSPATIAL Intl. Conf. Adv. Geographic Information Systems (GIS)*, 2019.

[122] Samik Banerjee, Lucas Magee, Dingkang Wang, Xu Li, Bing-Xing Huo, Jaikishan Jayakumar, Katherine Matho, Meng-Kuan Lin, Keerthi Ram, Mohanasankar Sivaprakasam, et al. Semantic segmentation of microscopic neuroanatomical data by combining topological priors with encoder–decoder deep networks. *Nature Machine Intelligence*, 2020.

[123] Michał Januszewski, Jörgen Kornfeld, Peter H Li, Art Pope, Tim Blakely, Larry Lindsey, Jeremy Maitin-Shepard, Mike Tyka, Winfried Denk, and Viren Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, 2018.

[124] Mustafa Gokhan Uzunbas, Chao Chen, and Dimitris Metaxas. An efficient conditional random field approach for automatic and interactive neuron segmentation. *Medical Image Analysis (MedIA)*, 2016.

[125] Ze Ye, Cong Chen, Changhe Yuan, and Chao Chen. Diverse multiple prediction on neuron image reconstruction. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2019.

[126] Uyeong Jang, Susmit Jha, and Somesh Jha. On the need for topology-aware generative models for manifold-based defenses. In *International Conference on Learning Representations (ICLR)*, 2020.

[127] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[128] Xiaoyu Zhang, Ajmal Mian, Rohit Gupta, Nazanin Rahnavard, and Mubarak Shah. Cassandra: Detecting trojaned networks from adversarial perturbations. *arXiv preprint arXiv:2007.14433*, 2020.

[129] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[130] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2013.

[131] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[132] Zhen Xiang, David J Miller, and George Kesidis. Revealing backdoors, post-training, in dnn classifiers via novel inference on optimized perturbations inducing group misclassification. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[133] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, 2018.

[134] Peter Bajcsy and Michael Majurski. Baseline pruning-based approach to trojan detection in neural networks. *arXiv preprint arXiv:2101.12016*, 2021.

[135] Edward Chou, Florian Tramèr, and Giancarlo Pellegrino. Sentinet: Detecting localized universal attacks against deep learning systems. In *IEEE Security and Privacy Workshops (SPW)*, 2020.

[136] Yansong Gao, Change Xu, Derui Wang, Shiping Chen, Damith C Ranasinghe, and Surya Nepal. Strip: A defence against trojan attacks on deep neural networks. In *Annual Computer Security Applications Conference*, 2019.

[137] Yuntao Liu, Yang Xie, and Ankur Srivastava. Neural trojans. In *IEEE International Conference on Computer Design (ICCD)*, 2017.

[138] Shiqing Ma and Yingqi Liu. Nic: Detecting adversarial samples with neural network invariant checking. In *Network and Distributed System Security (NDSS)*, 2019.

[139] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *SafeAI@ AAAI*, 2019.

[140] Guangyu Shen, Yingqi Liu, Guanhong Tao, Shengwei An, Qiuling Xu, Siyuan Cheng, Shiqing Ma, and Xiangyu Zhang. Backdoor scanning for deep neural networks through k-arm optimization. In *International Conference on Machine Learning (ICML)*, 2021.

[141] Mingjie Sun, Siddhant Agarwal, and J Zico Kolter. Poisoned classifiers are not only backdoored, they are fundamentally broken. *arXiv preprint arXiv:2010.09080*, 2020.

[142] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in neural information processing systems (NeurIPS)*, 2018.

[143] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[144] Soheil Kolouri, Aniruddha Saha, Hamed Pirsiavash, and Heiko Hoffmann. Universal litmus patterns: Revealing backdoor attacks in cnns. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020.

[145] Ren Wang, Gaoyuan Zhang, Sijia Liu, Pin-Yu Chen, Jinjun Xiong, and Meng Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *European Conference on Computer Vision (ECCV)*, 2020.

[146] Todd Huster and Emmanuel Ekwedike. Top: Backdoor detection in neural networks via transferability of perturbation. *arXiv preprint arXiv:2103.10274*, 2021.

[147] Songzhu Zheng, Yikai Zhang, Hubert Wagner, Mayank Goswami, and Chao Chen. Topological detection of trojaned neural networks. In *Advances in neural information processing systems (NeurIPS)*, 2021.

[148] Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems (NeurIPS)*, 2011.

[149] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, 2016.

[150] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems (NeurIPS)*, 2017.

[151] Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. Confidence-aware learning for deep neural networks. In *International Conference on Machine Learning (ICML)*, 2020.

[152] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. In *Advances in neural information processing systems (NeurIPS)*, 2012.

[153] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

[154] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. In *Advances in neural information processing systems (NeurIPS)*, 2016.

[155] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2015.

[156] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[157] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems (NeurIPS)*, 2017.

[158] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *International Conference on Computer Vision (ICCV)*, 2017.

[159] Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *European Conference on Computer Vision (ECCV)*, 2018.

[160] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete & Computational Geometry*, 2005.

[161] James R Munkres. *Elements of algebraic topology*. CRC press, 2018.

[162] Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *European Workshop on Computational Geometry*, 2011.

[163] Hubert Wagner, Chao Chen, and Erald Vuçini. Efficient computation of persistent homology for cubical data. In *Topological methods in data analysis and visualization II*. Springer, 2012.

[164] David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have l p-stable persistence. *Foundations of computational mathematics*, 2010.

[165] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 2011.

[166] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics (JEA)*, 2017.

[167] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 2007.

[168] Ignacio Arganda-Carreras, Srinivas C Turaga, Daniel R Berger, Dan Cireşan, Alessandro Giusti, Luca M Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M Buhmann, et al. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in neuroanatomy*, 2015.

[169] I Arganda-Carreras, HS Seung, A Vishwanathan, and D Berger. 3d segmentation of neurites in em images challenge-isbi, 2013.

[170] Qin Zou, Yu Cao, Qingquan Li, Qingzhou Mao, and Song Wang. Cracktree: Automatic crack detection from pavement images. *Pattern Recognition Letters (PRL)*, 2012.

[171] Volodymyr Mnih. *Machine learning for aerial image labeling*. University of Toronto (Canada), 2013.

[172] Joes Staal, Michael D Abràmoff, Meindert Niemeijer, Max A Viergever, and Bram Van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging (TMI)*, 2004.

[173] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2014.

[174] Tamal K. Dey and Yusu Wang. *Computational Topology for Data Analysis*. Cambridge University Press, 2021.

[175] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations (ICLR)*, 2019.

[176] Karan Sikka, Indranil Sur, Susmit Jha, Anirban Roy, and Ajay Divakaran. Detecting trojaned dnns using counterfactual attributions. *arXiv preprint arXiv:2012.02275*, 2020.

[177] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[178] Meir Barzohar and David B Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1996.

[179] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, CV Jawahar, and Manohar Paluri. Improved road connectivity by joint learning of orientation and segmentation. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2019.

[180] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018.

[181] Jan D Wegner, Javier A Montoya-Zegarra, and Konrad Schindler. A higher-order crf model for road network extraction. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2013.

[182] James Biagioni and Jakob Eriksson. Inferring road maps from global positioning system traces: Survey and comparative evaluation. *Transportation research record*, 2012.

[183] Christian Wiedemann, Christian Heipke, Helmut Mayer, and Olivier Jamet. Empirical evaluation of automatically extracted road axes. *Empirical evaluation techniques in computer vision*, 1998.

[184] Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *International Conference on Computer Vision (ICCV)*, 2017.

[185] Subeesh Vasu, Mateusz Kozinski, Leonardo Citraro, and Pascal Fua. Topoal: An adversarial learning approach for topology-aware road segmentation. In *European Conference on Computer Vision (ECCV)*, 2020.

[186] Agata Mosinska, Mateusz Koziński, and Pascal Fua. Joint segmentation and path classification of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[187] Xiaofei Yang, Xutao Li, Yunming Ye, Raymond YK Lau, Xiaofeng Zhang, and Xiaohui Huang. Road detection and centerline extraction via deep recurrent convolutional neural network u-net. *IEEE Transactions on Geoscience and Remote Sensing*, 2019.

[188] Jiaqi Yang, Xiaoling Hu, Chao Chen, and Chialing Tsai. A topological-attention convlstm network and its application to em images. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2021.

[189] Viren Jain, Benjamin Bollmann, Mark Richardson, Daniel R Berger, Moritz N Helmstaedter, Kevin L Briggman, Winfried Denk, Jared B Bowden, John M Mendenhall, Wickliffe C Abraham, et al. Boundary learning by optimization with topological constraints. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2010.

[190] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, 2002.

[191] T Yung Kong and Azriel Rosenfeld. Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing*, 1989.

[192] Gilles Bertrand and Grégoire Malandain. A new characterization of three-dimensional simple points. *Pattern Recognition Letters (PRL)*, 1994.

[193] Azriel Rosenfeld, T Yung Kong, and Akira Nakamura. Topology-preserving deformations of two-valued digital pictures. *Graphical Models and Image Processing*, 1998.

[194] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2018.

[195] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. Deepglobe 2018: A challenge to parse the earth through satellite images. In *CVPR Workshops*, 2018.

[196] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2016.

[197] Yong-Qiang Tan, Shang-Hua Gao, Xuan-Yi Li, Ming-Ming Cheng, and Bo Ren. Vecroad: Point-based iterative graph exploration for road graphs extraction. In *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2020.

[198] Zhenhua Xu, Yuxiang Sun, and Ming Liu. icurb: Imitation learning-based detection of road curbs using aerial images for autonomous driving. *IEEE Robotics and Automation Letters (RAL)*, 2021.

[199] R. Forman. Morse theory for cell complexes. *Advances in mathematics*, 1998.

[200] Thierry Sousbie. The persistent cosmic web and its filamentary structure–i. theory and implementation. *Monthly Notices of the Royal Astronomical Society*, 2011.

[201] Tamal K. Dey, Jiayuan Wang, and Yusu Wang. Graph reconstruction by discrete morse theory. In *Symposium on Computational Geometry (SoCG)*, 2018.

[202] U. Bauer, C. Lange, and M. Wardetzky. Optimal topological simplification of discrete functions on surfaces. *Discr. Comput. Geom.*, 2012.

[203] L Soler, A Hostettler, V Agnus, A Charnoz, J Fasquel, J Moreau, A Osswald, M Bouhadjar, and J Marescaux. 3d image reconstruction for comparison of algorithm database: a patient-specific anatomical and medical image database. *IRCAD, Strasbourg, France, Tech. Rep*, 2010.

[204] Tao Zeng, Bian Wu, and Shuiwang Ji. Deepem3d: approaching human-level performance on 3d anisotropic em image segmentation. *Bioinformatics*, 2017.

[205] Samuel Budd, Emma C Robinson, and Bernhard Kainz. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical Image Analysis (MedIA)*, 2021.

[206] Haotao Wang, Tianlong Chen, Zhangyang Wang, and Kede Ma. Troubleshooting image segmentation models with human-in-the-loop. *Machine Learning*, 2022.

[207] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[208] Samuel G Armato III, Geoffrey McLennan, Luc Bidaut, Michael F McNitt-Gray, Charles R Meyer, Anthony P Reeves, Binsheng Zhao, Denise R Aberle, Claudia I Henschke, Eric A Hoffman, et al. The lung image database

consortium (lidc) and image database resource initiative (idri): a completed reference database of lung nodules on ct scans. *Medical physics*, 2011.

[209] Kenneth Clark, Bruce Vendt, Kirk Smith, John Freymann, Justin Kirby, Paul Koppel, Stephen Moore, Stanley Phillips, David Maffitt, Michael Pringle, et al. The cancer imaging archive (tcia): maintaining and operating a public information repository. *Journal of digital imaging*, 2013.

[210] Philipp J Schubert, Sven Dorkenwald, Michał Januszewski, Viren Jain, and Joergen Kornfeld. Learning cellular morphology with neural networks. *Nature communications*, 2019.

[211] T Kasahara, A Takata, TM Kato, M Kubota-Sakashita, T Sawada, A Kakita, H Mizukami, D Kaneda, K Ozawa, and T Kato. Depression-like episodes in mice harboring mtdna deletions in paraventricular thalamus. *Molecular psychiatry*, 2016.

[212] Massimo Zeviani and Stefano Di Donato. Mitochondrial disorders. *Brain*, 2004.

[213] Christopher J Peddie and Lucy M Collinson. Exploring the third dimension: volume electron microscopy comes of age. *Micron*, 2014.

[214] Benjamin Titze and Christel Genoud. Volume scanning electron microscopy for imaging biological ultrastructure. *Biology of the Cell*, 2016.

[215] Larissa Heinrich, Davis Bennett, David Ackerman, Woohyun Park, John Bogovic, Nils Eckstein, Alyson Petruncio, Jody Clements, Song Pang, C Shan Xu, et al. Whole-cell organelle segmentation in volume electron microscopy. *Nature*, 2021.

[216] Manca Žerovnik Mekuč, Ciril Bohak, Samo Hudoklin, Byeong Hak Kim, Min Young Kim, Matija Marolt, et al. Automatic segmentation of mitochondria and endolysosomes in volumetric electron microscopy data. *Computers in biology and medicine*, 2020.

[217] Manuel Berning, Kevin M Boergens, and Moritz Helmstaedter. Segem: efficient image analysis for high-resolution connectomics. *Neuron*, 2015.

[218] Donglai Wei, Zudi Lin, Daniel Franco-Barranco, Nils Wendt, Xingyu Liu, Wenjie Yin, Xin Huang, Aarush Gupta, Won-Dong Jang, Xueying Wang,

et al. Mitoem dataset: Large-scale 3d mitochondria instance segmentation from em images. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2020.

[219] Xiaoling Hu, Xiao Lin, Michael Cogswell, Yi Yao, Susmit Jha, and Chao Chen. Trigger hunting with a topological prior for trojan detection. In *International Conference on Learning Representations (ICLR)*, 2022.

[220] Xiaoling Hu. Structure-aware image segmentation with homotopy warping. In *Advances in neural information processing systems (NeurIPS)*, 2022.

[221] Xiaoling Hu, Xiao Chen, Yikang Liu, Eric Z Chen, Terrence Chen, and Shanhui Sun. Deep statistic shape model for myocardium segmentation. *arXiv preprint arXiv:2207.10607*, 2022.

[222] Jiaqi Yang, Xiaoling Hu, Chao Chen, and Chialing Tsai. 3d topology-preserving segmentation with compound multi-slice representation. In *International Symposium on Biomedical Imaging (ISBI)*, 2021.

[223] Saumya Gupta, Xiaoling Hu, James Kaan, Michael Jin, Mutshipay Mpoy, Katherine Chung, Gagandeep Singh, Mary Saltz, Tahsin Kurc, Joel Saltz, et al. Learning topological interactions for multi-class medical image segmentation. In *European Conference on Computer Vision (ECCV)*, 2022.

[224] Saumya Gupta, Yikai Zhang, Xiaoling Hu, Prateek Prasanna, and Chao Chen. Topology-aware uncertainty for image segmentation. *arXiv preprint arXiv:2306.05671*, 2023.

[225] Chen Li, Xiaoling Hu, and Chao Chen. Confidence estimation using unlabeled data. In *International Conference on Learning Representations (ICLR)*, 2023.

[226] Chen Li, Xiaoling Hu, Shahira Abousamra, and Chao Chen. Calibrating uncertainty for semi-supervised crowd counting. In *International Conference on Computer Vision (ICCV)*, 2023.

[227] Aishik Konwer, Xiaoling Hu, Xuan Xu, Joseph Bae, Chao Chen, and Prateek Prasanna. Enhancing modality-agnostic representations via meta-learning for brain tumor segmentation. In *International Conference on Computer Vision (ICCV)*, 2023.