GraphMorph: Tubular Structure Extraction by Morphing Predicted Graphs

Zhao Zhang[1,5] **Ziwei Zhao**[2] **Dong Wang**[2] **Liwei Wang**[3,4,✉]
[1]Center for Data Science, Peking University [2]Yizhun Medical AI Co., Ltd
[3]State Key Laboratory of General Artificial Intelligence,
School of Intelligence Science and Technology, Peking University
[4]Center for Machine Learning Research, Peking University
[5]Pazhou Laboratory (Huangpu), Guangzhou, Guangdong, China
zhangzh@stu.pku.edu.cn ziwei.zhao@yizhun-ai.com
dong.wang@yizhun-ai.com wanglw@pku.edu.cn

Abstract

Accurately restoring topology is both challenging and crucial in tubular structure extraction tasks, such as blood vessel segmentation and road network extraction. Diverging from traditional approaches based on pixel-level classification, our proposed method, named GraphMorph, focuses on branch-level features of tubular structures to achieve more topologically accurate predictions. GraphMorph comprises two main components: a Graph Decoder and a Morph Module. Utilizing multi-scale features extracted from an image patch by the segmentation network, the Graph Decoder facilitates the learning of branch-level features and generates a graph that accurately represents the tubular structure in this patch. The Morph Module processes two primary inputs: the graph and the centerline probability map, provided by the Graph Decoder and the segmentation network, respectively. Employing a novel $\mathrm{SkeletonDijkstra}$ algorithm, the Morph Module produces a centerline mask that aligns with the predicted graph. Furthermore, we observe that employing centerline masks predicted by GraphMorph significantly reduces false positives in the segmentation task, which is achieved by a simple yet effective post-processing strategy. The efficacy of our method in the centerline extraction and segmentation tasks has been substantiated through experimental evaluations across various datasets. Source code will be released soon.

1 Introduction

Extraction of tubular structures is an essential step in many computer vision tasks [39, 13, 1, 27]. In medical applications, accurate segmentation of retinal vessels can provide crucial insights into various cardiovascular and ophthalmologic diseases [8]. In the field of urban planning and geographic information systems, the precise extraction of road networks aids in traffic management, urban development, and emergency response planning [28]. Existing deep learning-based methods model tubular structure extraction as a pixel-level classification task [34] or point set prediction task [40], without explicitly predicting the topological structures. To focus more on topology, some advanced methods design novel backbones or modules [36, 23, 33], or introduce new loss functions from the topological perspective [14, 37, 25]. However, they are still limited to the framework of pixel-level prediction.

We argue that most pixel-level frameworks have not effectively exploited the nature of tubular structures, which are inherently composed of several branches that are interconnected in complex ways. Specifically, pixel-level loss functions, like softDice Loss [26] and Focal Loss [20], struggle with subtle inaccuracies and are particularly ineffective at addressing complex topological errors.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-2187

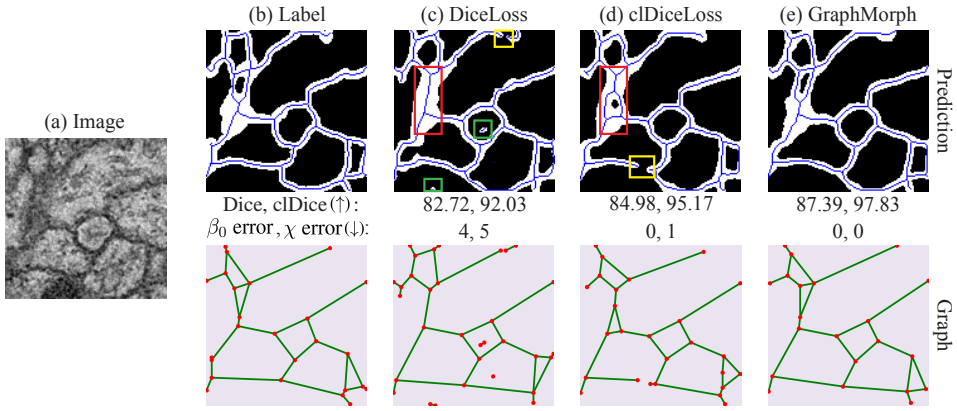


Figure 1: Illustrating the impact of topological feature utilization on segmentation accuracy. **(a)** An input neuron image. **Column (b)** Ground truth with segmented membranes (white) and its centerline (blue lines); the constructed graph (nodes in red, edges in green). **Column (c)** and **(d)** Predictions of two methods [26, 37] without explicit topological learning, highlighting broken branches (false negatives in yellow), redundant branches (false positives in green), and topological errors (in red). **Column (e)** Our GraphMorph guarantees topological accuracy by learning explicit branch-level features. Details of skeletonization and graph construction are given in Appendix B. Evaluation metrics: Dice and clDice (higher is better), β_0 error and χ error (lower is better).

Under the pixel-level frameworks, despite attempts to pay more attention to fine branches [37, 25] or topological features [14, 33], they still struggle with fully capturing the complex topological nature of tubular structures. We demonstrate this deficiency by providing an example in Figure 1. For a systematic understanding of the issues of pixel-level frameworks, we summarize them into three categories: (1) Broken branches or false negatives (FNs). (2) Redundant branches or false positives (FPs). (3) Topological errors (TEs). Therefore, understanding tubular structure extraction solely from a pixel-level perspective is fundamentally flawed.

Recognizing these limitations, we shift our focus to **branch-level features**, which are more essential for accurately capturing the nuances of tubular structures. Any complex tubular structure can be broken down into several branches, which distinguishes it from non-tubular objects. This perspective inspires us to extract tubular structures in two steps: (1) predicting the location of the two endpoints of each branch; (2) finding the optimal path between the two endpoints of each branch. Such a solution is intuitively aligned with human perception, and able to offer several advantages. Specifically, if the endpoints of branches are accurately predicted in the first step, redundant branches (FPs) are then potentially reduced; and the second step ensures that there is a path connecting the two endpoints of each branch, so that broken branches (FNs) and TEs are effectively suppressed. Besides, learning branch-level features during training elevates the model's focus on topology, which implicitly improves topological accuracy.

To effectively utilize branch-level features of tubular structures, we propose GraphMorph, a pipeline for obtaining topologically accurate centerline masks. GraphMorph consists of a Graph Decoder and a training-free Morph Module, which corresponds to the two steps of our solution respectively. The Graph Decoder, given multi-scale features extracted from an image patch by a segmentation network, predicts the graph G of the tubular structure. G is defined by a node set $V = \{(x_i, y_i)\}_{i=1}^{N}$, which contains the coordinates of all critical points vital for maintaining topology, and an adjacency matrix $A \in \{0, 1\}^{N \times N}$, which encodes the connectivity among nodes. Each pair of connected nodes in the graph corresponds to two endpoints of a branch of the tubular structure, thus the Graph Decoder takes full advantage of branch-level features through this graph representation. Technically, the prediction of V is addressed as a set prediction problem, solvable by our modified version of Deformable DETR [49]. To efficiently obtain the adjacency matrix A, we design a lightweight link prediction module that capitalizes on the extracted node features. Concretely, since the number of nodes in each tubular structure may be different, we generate linear weights and biases dynamically conditioned on node features, and the adjacency list of each node is obtained from its corresponding linear parameters (See Figure 2).

The Morph Module, a core contribution of this work, is intended to obtain topologically accurate centerline masks. While studies in the image-to-graph task [38, 32] also utilize graph representation,

they struggle to directly obtain accurate centerline masks due to the curved nature of tubular objects. In contrast, our Morph Module generates topologically accurate centerline masks via a novel $\mathrm{SkeletonDijkstra}$ algorithm. Specifically, a centerline probability map P_m, together with the graph G, output by the segmentation network and the Graph Decoder respectively, serve as the input to the Morph Module. Afterwards, considering the skeleton property of centerlines, our $\mathrm{SkeletonDijkstra}$ algorithm finds the optimal path between each pair of connected nodes. In particular, during path finding from the start point to the end point, we always restrict the path to a single pixel width to satisfy the skeleton property of centerlines. Consequently, the topology of the resulting centerline mask is guaranteed by G, leading to a reduction in TEs. This method also minimizes the occurrence of broken branches (FNs) and redundant branches (FPs), which is a significant improvement over direct pixel-level operations on P_m, such as thresholding.

We conduct the experiments by beginning with the **centerline extraction** task to verify the effects of the two components of GraphMorph. Experimentally, serveing as an auxiliary training module to learn the graph representation, the Graph Decoder enhances the segmentation network's focus on branch-level features, thus both volumetric metrics and topological metrics are boosted. Furthermore, employing the Morph Module at inference stage considerably improves topological metrics. For the **segmentation** task, we develop a streamlined post-processing strategy to refine segmentation masks via the topologically accurate centerline masks output by the Morph Module, significantly suppressing false positives of segmentation results. To verify the effectiveness of GraphMorph and the post-processing strategy, we conduct extensive experiments across four typical tubular structure extraction datasets. We have applied our methodology on three powerful backbones and achieved consistent improvements in all metrics. Moreover, compared with the state-of-the-art methods, our approach achieves the best results across all datasets.

In a nutshell, our contributions can be summarized as the following: (1) We introduce GraphMorph, an innovative framework specifically tailored for tubular structure extraction. Based on the proposed Graph Decoder and Morph Module, the branch-level features are fully exploited and the topologically accurate centerline masks are derived naturally. (2) For the segmentation task, an efficient post-processing strategy significantly suppresses false positives via the centerline masks predicted by GraphMorph, ensuring that the segmentation results are more closely aligned with the predicted graphs. (3) Experimental results on three medical datasets and one road dataset underscore the effectiveness of our method. For both centerline extraction and segmentation tasks, GraphMorph has achieved remarkable improvements across all metrics, especially in topological metrics.

2 Related Work

Image segmentation of tubular structures. Deep learning-based methods have achieved impressive results in segmentation tasks [21, 34, 5]. To further enhance the segmentation performance of tubular structures, novel network architectures [16, 22, 36, 41, 23, 45, 40, 33] and topology-preserving loss functions [14, 29, 37, 25, 33] have been proposed. For example, in terms of network architecture, DSCNet [33] utilizes dynamic snake convolution to capture fine and tortuous local features; PointScatter [40] explores the point set representation of tubular structures and introduces a novel greedy-based region-wise bipartite matching algorithm to improve training efficiency. In terms of loss functions, clDice [37] proposes a differentiable soft skeletonization method and achieves loss calculation at centerline level, which implicitly helps model focus more on the fine branches; TopoLoss [14] and TCLoss [33] measure the topological similarity of the ground truth and the prediction via persistent homology. Despite these advancements, all of the above methods are still confined to the framework of pixel-level classification and can not entirely overcome their inherent limitations. Our method attempts to morph the predicted graphs of tubular structures to let the network focus more on branch-level features, thus ensuring the topological accuracy of predictions.

Image to graph. There are two mainstream subtasks in this area: road network graph detection [11, 43, 38, 44, 12] and scene graph generation [17, 19]. These tasks usually entail detecting key components as nodes (i.e., key points in roads, objects in scenes) and determining their interrelations as edges (i.e., connectivity in roads, interactions in scenes). Our work differs from these approaches in three ways. Firstly, we use only junctions and endpoints as nodes, which allows for explicit semantic characterization of nodes in our graph representation, unlike road network detection tasks where path points may also be regarded as nodes. Secondly, considering the curved nature of tubular objects, we propose Morph Module to obtain topologically accurate centerline masks, a goal that is not addressed

 https://doi.org/10.52202/079017-2187

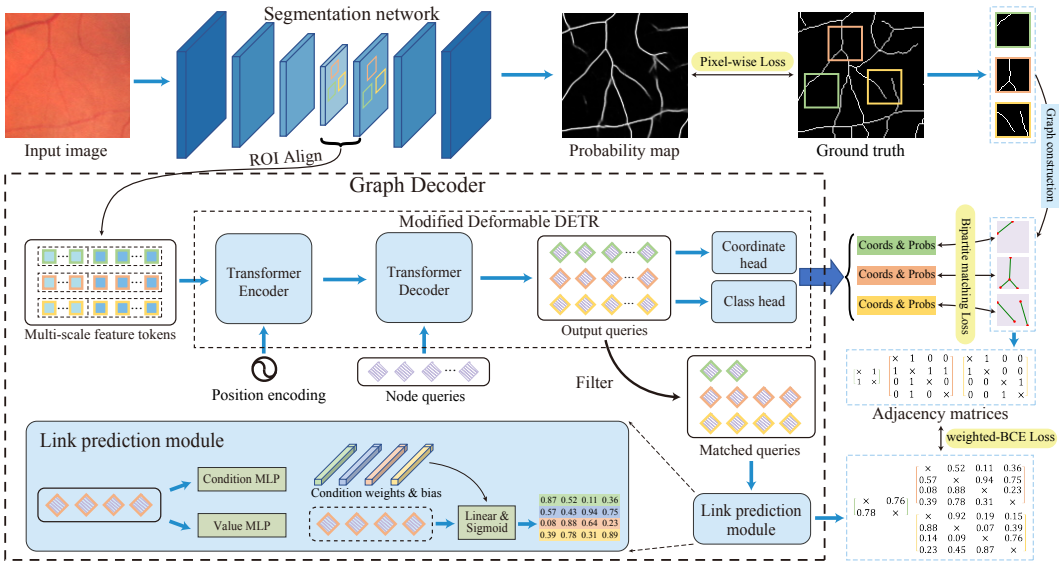


Figure 2: Overview of the training process. Given an image, the segmentation network outputs a probability map of the centerline or segmentation and produces multi-scale feature maps. Then, R regions of interest (ROIs) are randomly sampled from the image, and their corresponding features are fed into the Graph Decoder, which predicts the nodes within these ROIs using a modified Deformable DETR and outputs the adjacency matrices utilizing the proposed link prediction module.

by these works. Finally, our dynamic link prediction module is time-efficient, compared with elaborate and time-consuming designs in these works, such as [rln]-token in RelationFormer [38]. For a clear understanding, we experimentally compare the differences between our approach and RelationFormer in Appendix C. These distinctions make our model not only time-efficient but also applicable to the task of tubular structure extraction with more complex topology.

3 Method

This section provides a detailed description of the training and inference procedures of GraphMorph. Figure 2 illustrates the training process of our approach, where the segmentation network and Graph Decoder are included. We detail these two components in Section 3.1 and 3.2, respectively. The training details are given in Section 3.3. Section 3.4 introduces the algorithmic flow of the Morph Module, followed by the inference processes for the centerline extraction and segmentation tasks.

3.1 Segmentation Network

The segmentation network processes an input image I with shape $\mathbb{H} \times \mathbb{W}$. It serves two purposes: (1) outputting a probability map of tubular structures; (2) providing multi-scale features for the Graph Decoder. For training efficiency, we randomly sample R regions of interest (ROIs) with size $H \times H$ in the feature maps ($R = 3$ in Figure 2 for illustration). The ROI is defined as any region containing centerline points. The adoption of ROIs brings two key benefits: it reduces the model's learning complexity due to simpler topological structures within each ROI, and improves training efficiency by decreasing the number of feature tokens processed in the transformer. Technically, We adopt ROI Align [10] to extract multi-scale ROI features. Note that the generality of GraphMorph allows it to be adapted to any type of segmentation network. In the experimental part, we validate the enhancement of GraphMorph on a variety of segmentation networks.

3.2 Graph Decoder

The Graph Decoder is intended to predict the graph for each ROI. Specifically, the modified Deformable DETR [49] is responsible for detecting the nodes, while the link prediction module handles predicting the connectivity among these nodes. In the following, we will dissect each component to elucidate their roles.

Modified Deformable DETR. We have made two adjustments to the standard Deformable DETR [49]. Firstly, since the targets are nodes with only 2-dimensional coordinates, we replace the original box head with a coordinate head that outputs 2-dimensional vectors. Secondly, considering the typically small size of ROIs, we reduce the number of layers in the transformer encoder to three while keep the decoder at six layers. Note that each ROI is treated as an independent sample and different ROIs will not interact with each other in the whole training process. Formally, the process of node prediction can be expressed as follows:

$$\widehat{F}^r = \mathrm{TransformerEncoder}\left(F^r, PE\right) \tag{1}$$

$$\widehat{Q}^r = \mathrm{TransformerDecoder}\left(\widehat{F}^r, Q\right) \tag{2}$$

$$\hat{s}^r = \mathrm{Sigmoid}\left(\mathrm{ClassHead}\left(\widehat{Q}^r\right)\right), \quad \hat{v}^r = \mathrm{Sigmoid}\left(\mathrm{CoordHead}\left(\widehat{Q}^r\right)\right) \tag{3}$$

where $r = 1, 2, ..., R$. $F^r \in \mathbb{R}^{L\times C}$ denotes the multi-scale features of the r-th ROI, and $Q \in \mathbb{R}^{K\times C}$ is the initial node queries, where L and K are the scale of feature maps and the number of node queries respectively. PE is the multi-scale sinusoidal positional encoding used in [49]. $\mathrm{ClassHead}$ is a single linear layer, and $\mathrm{CoordHead}$ is a 3-layer multilayer perceptron (MLP). $\hat{s}^r \in \mathbb{R}^K$ and $\hat{v}^r \in \mathbb{R}^{K\times 2}$ are the classification scores and coordinates of the nodes for the r-th ROI respectively.

Link prediction module. Since the number of nodes for each ROI may be different, we design a dynamic module generating linear weights and biases conditioned on node features to directly predict the adjacency matrix A. For the r-th ROI sample, $\widehat{Q}^r \in \mathbb{R}^{K\times C}$ is the output queries of the Transformer decoder. The queries matched with the ground truth nodes (the number is denoted as P_r) during the bipartite matching process are preserved, and the rest queries are filtered out. We denote the kept queries as $\widetilde{Q}^r \in \mathbb{R}^{P_r\times C}$. As depicted in Figure 2, the matched queries $\widetilde{Q}^r$ will be fed into two MLPs. The $\mathrm{ConditionMLP}$ generates a $(C+1)$-dimensional vector for each matched query, which serves as the weights and biases of the condition linear layer. The $\mathrm{ValueMLP}$ maps the queries to a value space. With the values as input, the condition linear layer of the p-th query generates the adjacency list of it. The process can be formulated as follows:

$$W_p^r = \mathrm{ConditionMLP}(\widetilde{Q}_p^r) \in \mathbb{R}^{C+1}, \quad V^r = \mathrm{ValueMLP}(\widetilde{Q}^r) \in \mathbb{R}^{P_r\times C} \tag{4}$$

$$\widetilde{A}_p^r = \mathrm{Sigmoid}([W_p^r]_{1:C} \cdot V^r + [W_p^r]_{C+1}) \in \mathbb{R}^{P_r} \tag{5}$$

$$\widetilde{A}^r = [(\widetilde{A}_1^r)^T, (\widetilde{A}_2^r)^T, ..., (\widetilde{A}_{P_r}^r)^T]^T \in \mathbb{R}^{P_r\times P_r} \tag{6}$$

where $p = 1, 2, ..., P_r$. Here, $\widetilde{Q}_p^r$ denotes the p-th item of $\widetilde{Q}^r$, and C is its dimension. W_p^r refers to the linear parameters conditioned on the p-th matched query. $\widetilde{A}_p^r$ represents the predicted adjacency list for the p-th matched query, and the concatenation of all lists forms the final adjacency matrix $\widetilde{A}^r$.

3.3 Training Details

Graph construction. To train the Graph Decoder, we represent the ground truth of each ROI as a graph (see Figure 2). The detailed graph construction process can be found in Appendix B.

Label assignment based on bipartite matching. Bipartite matching is widely used in solving set prediction problems [2, 49, 40]. As in [49], we first calculate the cost between the predicted and ground truth nodes. The predicted nodes are denoted as $\hat{y} = \{(\hat{s}_k, \hat{v}_k)\}_{k=1}^K$, where we omit the index r of the ROI sample for simplicity. Under the general assumption that K is larger than the number of ground truth nodes P_r, thus we pad the set of ground truth nodes with $\varnothing$ (no node) to achieve a size of K. The ground truth set can be denoted as $y = \{(c_i, v_i)\}_{i=1}^K$, where c_i is the target class label and $v_i \in [0, 1]^2$ is the coordinate of the node. For a permutation $\sigma \in \mathfrak{S}_K$, where $\hat{y}_{\sigma(i)}$ is assigned to y_i $(i = 1, 2, .., K)$, we define the cost between y_i and $\hat{y}_{\sigma(i)}$ as:

$$\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = \lambda_{\text{class}} \cdot \mathbb{1}_{\{c_i\neq\varnothing\}}\mathcal{L}_{\text{class}}(\hat{s}_{\sigma(i)}) + \lambda_{\text{coord}} \cdot \mathbb{1}_{\{c_i\neq\varnothing\}}\mathcal{L}_{\text{coord}}(v_i, \hat{v}_{\sigma(i)}) \tag{7}$$

where λ_{class} and λ_{coord} are hyperparameters. $\mathcal{L}_{\text{class}}(\hat{s}_{\sigma(i)}) = \mathcal{L}_{\text{focal}}(\hat{s}_{\sigma(i)}, 1) - \mathcal{L}_{\text{focal}}(\hat{s}_{\sigma(i)}, 0)$, $\mathcal{L}_{\text{focal}}(s, c)$ is defined as $-\alpha \cdot (1-s)^\gamma \log(s)$ if $c = 1$, and $-(1-\alpha) \cdot s^\gamma \log(1-s)$ if $c = 0$, where α and γ are hyperparameters. $\mathcal{L}_{\text{coord}}$ is commonly used ℓ_1 loss. The optimal $\hat{\sigma}$ is defined as

$$\hat{\sigma} = \underset{\sigma\in\mathfrak{S}_K}{\arg\min} \sum_{i=1}^{K} \mathcal{L}_{\text{match}}\left(y_i, \hat{y}_{\sigma(i)}\right) \tag{8}$$

https://doi.org/10.52202/079017-2187

This optimal assignment can be efficiently obtained by Hungarian algorithm [18].

Loss functions. To train the Graph Decoder, the overall loss function is comprised of three components: pixel-wise loss $\mathcal{L}_{\text{Pixel}}$ between the probability map and the ground truth binary mask (in this work, we use softDice [26] and clDice [37]), Hungarian bipartite matching loss $\mathcal{L}_{\text{Hungarian}}$ between the predicted and ground truth nodes, weighted-BCE loss $\mathcal{L}_{\text{Adjacency}}$ between the predicted and ground truth adjacency matrices of the matched queries. For an image with R ROI samples, the last two loss functions are defined as:

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{r=1}^{R} \sum_{i=1}^{K} \left[\lambda_{\text{class}} \cdot \mathcal{L}_{\text{focal}}(\hat{s}^r_{\hat{\sigma}(i)}, c^r_i) + \lambda_{\text{coord}} \cdot \mathbb{1}_{\{c^r_i \neq \varnothing\}} \mathcal{L}_{\text{coord}} \left(\hat{v}^r_{\hat{\sigma}(i)}, v^r_i \right) \right], \quad (9)$$

$$\mathcal{L}_{\text{Adjacency}}(y, \hat{y}) = \sum_{r=1}^{R} \{ \frac{0.5}{N_{\text{pos}}} \sum_{i \neq j}^{P_r} \sum_{j=1}^{P_r} (A^r_{ij} \log \widetilde{A}^r_{ij}) + \frac{0.5}{N_{\text{neg}}} \sum_{i \neq j}^{P_r} \sum_{j=1}^{P_r} [(1 - A^r_{ij}) \log(1 - \widetilde{A}^r_{ij})] \}, \quad (10)$$

where N_{pos} is the total number of positive locations in ground truth $\{A^r\}_{r=1}^{R}$, and N_{neg} is the total number of negative locations. Thus, the overall loss function is $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Pixel}} + \mathcal{L}_{\text{Hungarian}} + \mathcal{L}_{\text{Adjacency}}$.

3.4 Morph Module and Inference

The Morph Module is used to get topologically accurate centerline masks, by morphing the predicted graphs from the Graph Decoder. In this subsection, we first introduce the Morph Module, followed by the inference processes for the centerline extraction and segmentation tasks.

Morph Module. We present the algorithmic flow in Algorithm 1. In particular, $G = \{V, E\}$ is the graph of an image patch (same size as an ROI sample), and P_m is the probability map of centerlines obtained from the segmentation network. We iterate over each edge and use our proposed $\mathrm{SkeletonDijkstra}$ algorithm to find the optimal path with minimum cost. The union of these paths forms the final centerline mask.

$\mathrm{SkeletonDijkstra}$ is modified from Dijkstra algorithm [7]. We have made two key adaptations for centerline extraction: (1) To restrict the path to a single pixel width, ensuring the property of the skeleton, we mandate that all path points, except for the start and end points, satisfy $N = 2$ (where N is the number of centerline points in its eight neighbours, see Appendix B). (2) To suppress potential false-positive edges from the Graph Decoder, we exclude the paths with an average cost exceeding a threshold p_{thresh}. These refinements optimize the algorithm to yield topologically accurate centerline masks. The detailed algorithmic flow of $\mathrm{SkeletonDijkstra}$ can be seen in Algorithm 2 in Appendix D.

Algorithm 1 Morph Module

Input: Node set V, Edge set E, Probability map P_m
Output: Centerline mask M

```
Initialize M as a zero matrix with the same size as P_m
Initialize C_m where C_m[i][j] = 1 − P_m[i][j] for each element
for all edges (u, v) in E do
    path ← SkeletonDijkstra(u, v, C_m, p_thresh)
    for all points p in path do
        Set M[p.x][p.y] = 1
    end for
end for
return M
```

Inference of centerline extraction. As depicted in Figure 3, the centerline extraction process begins with generating a centerline probability map via the segmentation network. Then, sliding window inference is employed across the entire image in Graph Decoder to obtain graphs for all split patches. Finally, the Morph Module produces the centerline mask for each patch, and the combination of these masks forms the complete centerline mask of the entire image.

Inference of segmentation. Since the segmentation probability map S_m can not be used directly by the Morph Module, we first threshold S_m to obtain segmentation mask S'_m and skeletonize it into a centerline mask P'_m. The distance from each pixel to the nearest centerline point in P'_m is calculated

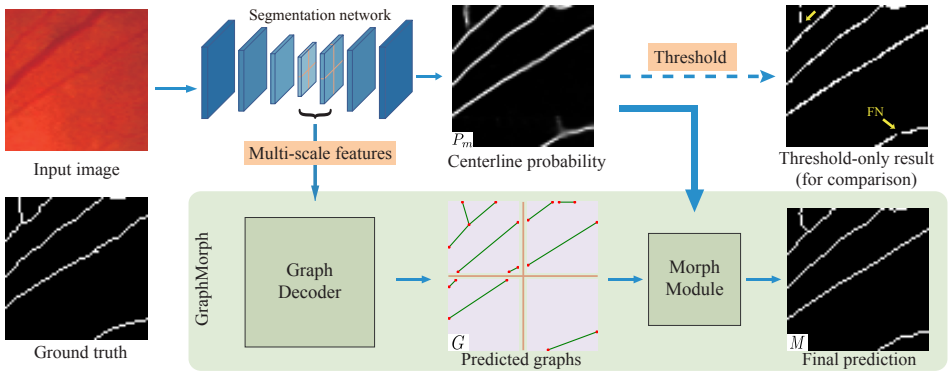


Figure 3: Inference process of centerline extraction. First, the segmentation network generates a centerline probability map P_m along with multi-scale image features. Subsequently, the Graph Decoder utilizes the image features to predict graphs G via sliding window inference. Finally, the Morph Module employs P_m to find the optimal path between each pair of connected nodes in G, resulting in a final centerline mask. This approach achieves higher topological accuracy compared to direct thresholding of P_m.

and normalized to create a distance map D. Then the centerline probability map P_m is obatined by $P_m = S_m \times (1 - D)$. Employing the Morph Module on P_m yields a topologically precise mask M. To suppress false positives in S'_m (especially isolated regions), a post-processing strategy is initiated from $M_0 = M \odot S'_m$. This strategy involves iteratively expanding M_0 within the boundaries of S'_m until stabilization. The stabilized mask M_T is then taken as the final output. This approach, as confirmed by experiments, effectively diminishes false positives and enhances topological accuracy. The above-mentioned soft skeletonization method (from S_m to P_m) and post-processing strategy introduce minimal time cost and are straightforward to implement, with detailes in Appendix E.1 and Appendix E.2.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate GraphMorph on three medical datasets and one road dataset. DRIVE [39] and STARE [13] are retinal vessel datasets commonly used in medical image segmentation. ISBI12 [1] contains 30 Electron Microscopy images to segment membranes. The Massachusetts Roads (MassRoad) dataset contains 1171 aerial images for road network extraction. We use the data splits for DRIVE and STARE provided in MMSegmentation [6]. For ISBI12, following previous works [35, 29], we split it into 15 images for training and 15 for testing. For MassRoad, we follow [40] to construct the training set, and the total 63 images of the official validation set and test set are used for testing.

Baselines. We adopt affluent baselines for comparison, including UNet [34], ResUNet [47], CS-Net [30], DC-UNet [22], TransUNet [4], DSCNet [33] and PointScatter [40]. Particularly, we use LinkNet34 [3] and D-Linknet34 [48] as baselines for the MassRoad dataset. In addition, we compare with TopoLoss [14], which is a topology-based loss function.

Metrics. For the centerline extraction task, we use Dice [50], Accuracy (ACC) and AUC as volumetric metrics. For robust evaluation, we give a tolerance of a 5-pixel region around the ground truth centerline mask following [9]. We compute topological metrics following [40], including the mean absolute errors of β_0, β_1 and the Euler characteristic. To compare fairly, we skeletonize the prediction before evaluation. For the segmentation task, we adopt Dice, clDice [37] and ACC as volumetric metrics and the same topological metrics as the centerline extraction task. Moreover, ARI (Adjusted Rand Index) [15] and VOI (Variation of Information) [24] are used to evaluate clustering similarity.

Implementation Details. For three medical datasets, we use randomly cropped 384×384 images for training. The size of ROI samples H is 32 and the stride of sliding window used in inference process is 30. For MassRoad dataset, the cropped size is 768×768, $H = 48$ and the stride is 45. For all experiments, we use 64 ROI samples per image ($R = 64$) to train the Graph Decoder, and the number of node queries in the modified Deformable DETR is set to 100 ($K = 100$). According to previous

https://doi.org/10.52202/079017-2187

experiences [38], the default hyperparameters used in loss functions are as follows: $\lambda_{\text{class}} = 0.2$, $\lambda_{\text{coord}} = 0.5$, $\alpha = 0.6$, $\gamma = 2$. We use $\alpha = 0.75$ for MassRoad due to the sparse nature of the road networks. For all types of segmentation networks, we use multi-scale features ranging from the lowest resolution to the $4\times$ downsampling of the original image as input of the Graph Decoder. More implementation details are introduced in Appendix A.

Table 1: Centerline extraction performance on four public datasets based on UNet.

Dataset	Method	Volumetric metrics (↑)			Topological metrics (↓)		
		Dice	AUC	ACC	β_0 error	β_1 error	χ error
DRIVE	softDice [26]	0.7353 ± 0.0127	0.9333 ± 0.0089	0.9768 ± 0.0013	2.169 ± 0.112	1.590 ± 0.107	2.537 ± 0.139
	PointScatter [40]	0.7381 ± 0.0133	0.9401 ± 0.0078	0.9775 ± 0.0013	3.259 ± 0.153	2.080 ± 0.120	3.500 ± 0.176
	softDice [26] + Graph Decoder	**0.7506 ± 0.0127**	**0.9481 ± 0.0082**	**0.9783 ± 0.0012**	1.552 ± 0.094	1.382 ± 0.106	1.899 ± 0.125
	softDice [26] + Graph Decoder + Morph Module	0.7496 ± 0.0118	/	0.9776 ± 0.0012	**0.555 ± 0.038**	**1.074 ± 0.073**	**0.893 ± 0.061**
ISBI12	softDice [26]	0.6428 ± 0.0104	0.8937 ± 0.0063	0.9737 ± 0.0013	4.045 ± 0.191	2.696 ± 0.112	4.294 ± 0.205
	Pointscatter [40]	0.6546 ± 0.0089	0.9104 ± 0.0057	**0.9747 ± 0.0013**	6.398 ± 0.277	3.156 ± 0.124	6.548 ± 0.290
	softDice [26] + Graph Decoder	0.6486 ± 0.0095	**0.9240 ± 0.0061**	0.9742 ± 0.0012	4.013 ± 0.179	2.732 ± 0.110	4.249 ± 0.193
	softDice [26] + Graph Decoder + Morph Module	**0.6687 ± 0.0092**	/	0.9742 ± 0.0014	**0.665 ± 0.049**	**1.207 ± 0.070**	**0.858 ± 0.059**
STARE	softDice [26]	0.7119 ± 0.0392	0.9290 ± 0.0283	0.9889 ± 0.0012	1.874 ± 0.139	1.209 ± 0.112	2.063 ± 0.162
	Pointscatter [40]	0.7224 ± 0.0414	0.9494 ± 0.0179	0.9896 ± 0.0012	2.080 ± 0.149	1.365 ± 0.116	2.213 ± 0.166
	softDice [26] + Graph Decoder	**0.7298 ± 0.0428**	**0.9506 ± 0.0208**	**0.9898 ± 0.0011**	1.467 ± 0.113	1.074 ± 0.104	1.654 ± 0.132
	softDice [26] + Graph Decoder + Morph Module	0.7291 ± 0.0387	/	0.9894 ± 0.0011	**0.482 ± 0.042**	**0.799 ± 0.077**	**0.653 ± 0.059**
MassRoad	softDice [26]	0.6339 ± 0.0169	0.9718 ± 0.0047	**0.9942 ± 0.0009**	1.672 ± 0.056	1.627 ± 0.087	1.968 ± 0.097
	Pointscatter [40]	**0.6405 ± 0.0149**	0.9694 ± 0.0042	**0.9942 ± 0.0009**	3.333 ± 0.124	1.553 ± 0.086	3.429 ± 0.149
	softDice [26] + Graph Decoder	0.6289 ± 0.0175	**0.9731 ± 0.0045**	0.9941 ± 0.0009	1.933 ± 0.065	1.729 ± 0.088	2.229 ± 0.105
	softDice [26] + Graph Decoder + Morph Module	0.6388 ± 0.0168	/	**0.9942 ± 0.0009**	**0.620 ± 0.021**	**1.355 ± 0.083**	**1.122 ± 0.075**

4.2 Main Results

We first verify the effectiveness of GraphMorph on the centerline extraction task. Then, considerable experiments are conducted on the more common segmentation task, demonstrating the powerful topological modelling capability of GraphMorph.

Centerline Extraction. In our experiments with UNet and softDice loss on four public datasets, detailed in Table 1, the inclusion of the Graph Decoder during training enables the network to learn branch-level features, leading to enhanced performance in both volumetric and topological metrics. During inference, the utilization of Morph Module results in a slight decrease in volumetric metrics; however, there is a notable enhancement in topological metrics, confirming that our network has effectively captured branch-level features of tubular structures. Overall, the combined use of the Graph Decoder and Morph Module showcases the ability to refine the segmentation network's performance, particularly in preserving the crucial topological characteristics. Our methods also beat previous SOTA Pointscatter [40] by a large margin..

Table 2: Segmentation performance based on different segmentation networks.

Dataset	Backbone	Method	Volumetric metrics (↑)			Distribution metrics		Topological metrics (↓)		
			Dice	clDice	ACC	ARI(↑)	VOI(↓)	β_0 error	β_1 error	χ error
DRIVE	UNet [34]	softDice [26]	0.8148 ± 0.0093	0.8128 ± 0.0169	0.9535 ± 0.0023	0.767 ± 0.011	0.348 ± 0.012	1.191 ± 0.069	1.078 ± 0.074	1.467 ± 0.083
		softDice+Ours	**0.8238 ± 0.0091**	**0.8278 ± 0.0166**	**0.9557 ± 0.0023**	**0.778 ± 0.011**	**0.336 ± 0.012**	**0.692 ± 0.047**	**0.932 ± 0.068**	**0.951 ± 0.062**
	ResUNet [47]	softDice [26]	0.8183 ± 0.0094	0.8183 ± 0.0178	0.9543 ± 0.0022	0.771 ± 0.011	0.342 ± 0.011	1.110 ± 0.066	1.059 ± 0.073	1.379 ± 0.079
		softDice+Ours	**0.8233 ± 0.0095**	**0.8273 ± 0.0172**	**0.9555 ± 0.0022**	**0.777 ± 0.011**	**0.336 ± 0.011**	**0.723 ± 0.047**	**0.986 ± 0.071**	**0.993 ± 0.063**
	CS-Net [30]	softDice [26]	0.8089 ± 0.0123	0.8073 ± 0.0185	0.9527 ± 0.0027	0.761 ± 0.014	0.350 ± 0.011	1.211 ± 0.072	1.096 ± 0.076	1.491 ± 0.085
		softDice+Ours	**0.8223 ± 0.0088**	**0.8253 ± 0.0171**	**0.9554 ± 0.0021**	**0.776 ± 0.010**	**0.336 ± 0.011**	**0.680 ± 0.044**	**0.990 ± 0.070**	**0.929 ± 0.059**
ISBI12	UNet [34]	softDice [26]	0.8043 ± 0.0092	0.9295 ± 0.0078	0.9146 ± 0.0060	0.653 ± 0.018	0.785 ± 0.040	0.569 ± 0.046	0.616 ± 0.047	0.738 ± 0.052
		softDice+Ours	**0.8216 ± 0.0091**	**0.9449 ± 0.0069**	**0.9211 ± 0.0057**	**0.678 ± 0.018**	**0.745 ± 0.039**	**0.361 ± 0.034**	**0.520 ± 0.043**	**0.488 ± 0.041**
	ResUNet [47]	softDice [26]	0.8061 ± 0.0093	0.9307 ± 0.0086	0.9153 ± 0.0055	0.655 ± 0.017	0.781 ± 0.037	0.572 ± 0.045	0.588 ± 0.048	0.710 ± 0.050
		softDice+Ours	**0.8166 ± 0.0105**	**0.9405 ± 0.0087**	**0.9194 ± 0.0055**	**0.671 ± 0.018**	**0.756 ± 0.037**	**0.395 ± 0.037**	**0.576 ± 0.046**	**0.518 ± 0.043**
	CS-Net [30]	softDice [26]	0.8163 ± 0.0118	0.9391 ± 0.0092	0.9194 ± 0.0075	0.671 ± 0.023	0.754 ± 0.049	0.451 ± 0.040	0.596 ± 0.046	0.622 ± 0.047
		softDice+Ours	**0.8282 ± 0.0118**	**0.9469 ± 0.0081**	**0.9243 ± 0.0068**	**0.690 ± 0.022**	**0.722 ± 0.045**	**0.342 ± 0.034**	**0.501 ± 0.043**	**0.452 ± 0.040**
STARE	UNet [34]	softDice [26]	0.8170 ± 0.0402	0.8526 ± 0.0306	0.9749 ± 0.0044	0.781 ± 0.042	0.276 ± 0.033	0.786 ± 0.064	0.653 ± 0.072	0.960 ± 0.079
		softDice+Ours	**0.8210 ± 0.0464**	**0.8578 ± 0.0372**	**0.9756 ± 0.0045**	**0.786 ± 0.049**	**0.271 ± 0.033**	**0.545 ± 0.046**	**0.618 ± 0.067**	**0.691 ± 0.059**
	ResUNet [47]	softDice [26]	0.7982 ± 0.0628	0.8343 ± 0.0512	0.9735 ± 0.0055	0.761 ± 0.065	0.282 ± 0.036	0.770 ± 0.067	0.707 ± 0.073	0.915 ± 0.079
		softDice+Ours	**0.8151 ± 0.0513**	**0.8522 ± 0.0404**	**0.9752 ± 0.0047**	**0.779 ± 0.053**	**0.273 ± 0.034**	**0.582 ± 0.054**	**0.623 ± 0.068**	**0.743 ± 0.065**
	CS-Net [30]	softDice [26]	0.7785 ± 0.0615	0.8173 ± 0.0474	0.9715 ± 0.0056	0.739 ± 0.063	0.294 ± 0.037	0.871 ± 0.071	0.803 ± 0.081	1.049 ± 0.087
		softDice+Ours	**0.7968 ± 0.0612**	**0.8351 ± 0.0483**	**0.9733 ± 0.0059**	**0.760 ± 0.064**	**0.282 ± 0.039**	**0.579 ± 0.049**	**0.685 ± 0.071**	**0.724 ± 0.062**
MassRoad	UNet [34]	softDice [26]	0.7808 ± 0.0146	0.8768 ± 0.0159	0.9780 ± 0.0036	0.750 ± 0.017	0.239 ± 0.033	0.479 ± 0.020	0.798 ± 0.076	0.777 ± 0.072
		softDice+Ours	**0.7849 ± 0.0139**	**0.8816 ± 0.0151**	**0.9783 ± 0.0035**	**0.754 ± 0.016**	**0.237 ± 0.033**	**0.386 ± 0.016**	**0.754 ± 0.076**	**0.672 ± 0.070**
	ResUNet [47]	softDice [26]	0.7730 ± 0.0152	0.8663 ± 0.0162	0.9773 ± 0.0036	0.742 ± 0.017	0.245 ± 0.033	0.799 ± 0.030	0.902 ± 0.078	1.089 ± 0.076
		softDice+Ours	**0.7755 ± 0.0150**	**0.8707 ± 0.0162**	**0.9775 ± 0.0036**	**0.744 ± 0.017**	**0.243 ± 0.033**	**0.587 ± 0.023**	**0.869 ± 0.078**	**0.869 ± 0.073**
	CS-Net [30]	softDice [26]	0.7770 ± 0.0147	0.8716 ± 0.0163	0.9779 ± 0.0035	0.746 ± 0.016	0.240 ± 0.032	0.487 ± 0.020	0.796 ± 0.075	0.784 ± 0.072
		softDice+Ours	**0.7789 ± 0.0149**	**0.8756 ± 0.0163**	**0.9779 ± 0.0035**	**0.748 ± 0.017**	**0.240 ± 0.033**	**0.399 ± 0.017**	**0.772 ± 0.076**	**0.682 ± 0.070**

Segmentation. In the initial phase of our segmentation experiments, we assessed the effectiveness of our method across different segmentation networks and datasets. As detailed in Table 2, enhancements were evident in various metrics for all dataset and network combinations. The improvements in

Table 3: Comparison with SOTA methods on the segmentation task. Best results are in bold; second-best are underlined. Our approach secures all leading scores and most secondary peaks.

Dataset	Backbone	Method	Volumetric metrics (↑)			Distribution metrics		Topological metrics (↓)		
			Dice	clDice	ACC	ARI(↑)	VOI(↓)	β_0 error	β_1 error	χ error
DRIVE	UNet	softDice [26]	0.8148 ± 0.0093	0.8128 ± 0.0169	0.9535 ± 0.0023	0.767 ± 0.011	0.348 ± 0.012	1.191 ± 0.069	1.078 ± 0.074	1.467 ± 0.083
	UNet	clDice [37]	0.8150 ± 0.0078	0.8322 ± 0.0163	0.9520 ± 0.0021	0.765 ± 0.009	0.357 ± 0.012	0.910 ± 0.056	0.998 ± 0.070	1.181 ± 0.071
	UNet	Pointscatter [40]	0.8155 ± 0.0081	0.8277 ± 0.0179	0.9525 ± 0.0020	0.766 ± 0.009	0.353 ± 0.010	1.360 ± 0.080	1.276 ± 0.083	1.663 ± 0.094
	UNet	TopoLoss [14]	0.8187 ± 0.0075	0.8194 ± 0.0160	0.9540 ± 0.0020	0.771 ± 0.009	0.345 ± 0.010	0.821 ± 0.050	0.997 ± 0.072	1.100 ± 0.067
	DSCNet [33]	softDice	0.8118 ± 0.0083	0.8107 ± 0.0172	0.9527 ± 0.0021	0.763 ± 0.010	0.352 ± 0.011	1.267 ± 0.075	1.110 ± 0.076	1.550 ± 0.087
	TransUNet [4]	softDice	0.8153 ± 0.0094	0.8139 ± 0.0182	0.9538 ± 0.0020	0.768 ± 0.010	0.344 ± 0.009	1.125 ± 0.066	1.184 ± 0.082	1.420 ± 0.082
	DC-UNet [22]	softDice	0.8086 ± 0.0103	0.8018 ± 0.0163	0.9526 ± 0.0024	0.760 ± 0.012	0.351 ± 0.011	1.227 ± 0.074	1.061 ± 0.074	1.499 ± 0.087
	UNet	softDice+Ours	**0.8238 ± 0.0091**	0.8278 ± 0.0166	**0.9557 ± 0.0023**	**0.778 ± 0.011**	**0.336 ± 0.012**	0.692 ± 0.047	0.932 ± 0.068	0.951 ± 0.062
	UNet	clDice+Ours	0.8168 ± 0.0076	**0.8467 ± 0.0146**	0.9520 ± 0.0021	0.767 ± 0.009	0.357 ± 0.012	**0.619 ± 0.043**	**0.924 ± 0.065**	**0.857 ± 0.056**
ISBI12	UNet	softDice [26]	0.8043 ± 0.0092	0.9295 ± 0.0078	0.9146 ± 0.0060	0.653 ± 0.018	0.785 ± 0.040	0.569 ± 0.046	0.616 ± 0.047	0.738 ± 0.052
	UNet	clDice [37]	0.8103 ± 0.0099	0.9353 ± 0.0084	0.9163 ± 0.0064	0.660 ± 0.020	0.775 ± 0.042	0.422 ± 0.038	0.563 ± 0.045	0.576 ± 0.043
	UNet	Pointscatter [40]	0.8192 ± 0.0101	0.9406 ± 0.0077	0.9189 ± 0.0063	0.672 ± 0.020	0.758 ± 0.042	0.456 ± 0.041	0.568 ± 0.046	0.587 ± 0.047
	UNet	TopoLoss [14]	0.8104 ± 0.0090	0.9324 ± 0.0074	0.9167 ± 0.0058	0.661 ± 0.017	0.773 ± 0.039	0.516 ± 0.041	0.642 ± 0.052	0.669 ± 0.049
	DSCNet [33]	softDice	0.8152 ± 0.0087	0.9366 ± 0.0078	0.9191 ± 0.0054	0.669 ± 0.016	0.757 ± 0.037	0.450 ± 0.040	0.567 ± 0.045	0.581 ± 0.044
	TransUNet [4]	softDice	0.8056 ± 0.0080	0.9289 ± 0.0075	0.9148 ± 0.0055	0.654 ± 0.016	0.784 ± 0.037	0.636 ± 0.049	0.576 ± 0.047	0.757 ± 0.053
	DC-UNet [22]	softDice	0.8150 ± 0.0089	0.9366 ± 0.0084	0.9196 ± 0.0063	0.671 ± 0.019	0.753 ± 0.043	0.511 ± 0.043	0.586 ± 0.046	0.652 ± 0.047
	UNet	softDice+Ours	0.8216 ± 0.0091	0.9449 ± 0.0069	0.9211 ± 0.0057	0.678 ± 0.018	0.745 ± 0.039	0.361 ± 0.034	**0.520 ± 0.043**	0.488 ± 0.041
	UNet	clDice+Ours	**0.8223 ± 0.0086**	**0.9459 ± 0.0066**	**0.9213 ± 0.0056**	**0.679 ± 0.017**	**0.744 ± 0.038**	**0.353 ± 0.034**	0.539 ± 0.043	**0.482 ± 0.040**
STARE	UNet	softDice [26]	0.8170 ± 0.0402	0.8526 ± 0.0306	0.9749 ± 0.0044	0.781 ± 0.042	0.276 ± 0.033	0.786 ± 0.064	0.653 ± 0.072	0.960 ± 0.079
	UNet	clDice [37]	0.8212 ± 0.0386	0.8579 ± 0.0319	0.9752 ± 0.0041	0.785 ± 0.040	0.276 ± 0.032	0.571 ± 0.049	0.629 ± 0.069	0.743 ± 0.065
	UNet	Pointscatter [40]	0.8171 ± 0.0395	0.8533 ± 0.0331	0.9743 ± 0.0041	0.780 ± 0.041	0.285 ± 0.031	0.844 ± 0.070	0.781 ± 0.080	0.997 ± 0.086
	UNet	TopoLoss [14]	0.8175 ± 0.0449	0.8506 ± 0.0339	0.9750 ± 0.0045	0.781 ± 0.047	0.276 ± 0.033	0.659 ± 0.056	0.615 ± 0.068	0.806 ± 0.069
	DSCNet [33]	softDice	0.7988 ± 0.0420	0.8341 ± 0.0348	0.9723 ± 0.0052	0.759 ± 0.045	0.296 ± 0.037	0.823 ± 0.068	0.707 ± 0.072	0.988 ± 0.080
	TransUNet [4]	softDice	0.8046 ± 0.0474	0.8428 ± 0.0370	0.9737 ± 0.0047	0.767 ± 0.049	0.284 ± 0.034	0.728 ± 0.061	0.723 ± 0.076	0.884 ± 0.078
	DC-UNet [22]	softDice	0.7936 ± 0.0547	0.8300 ± 0.0426	0.9728 ± 0.0052	0.755 ± 0.057	0.288 ± 0.034	0.834 ± 0.071	0.721 ± 0.075	0.975 ± 0.082
	UNet	softDice+Ours	0.8210 ± 0.0464	0.8578 ± 0.0372	0.9756 ± 0.0045	0.786 ± 0.049	**0.271 ± 0.033**	0.545 ± 0.046	0.618 ± 0.067	0.691 ± 0.059
	UNet	clDice+Ours	**0.8283 ± 0.0371**	**0.8747 ± 0.0284**	**0.9757 ± 0.0040**	**0.792 ± 0.039**	0.274 ± 0.032	**0.450 ± 0.042**	**0.582 ± 0.065**	**0.598 ± 0.055**
MassRoad	UNet	softDice [26]	0.7808 ± 0.0146	0.8768 ± 0.0159	0.9780 ± 0.0036	0.750 ± 0.017	0.239 ± 0.033	0.479 ± 0.020	0.798 ± 0.076	0.777 ± 0.072
	UNet	clDice [37]	0.7788 ± 0.0143	0.8773 ± 0.0156	0.9775 ± 0.0037	0.747 ± 0.016	0.244 ± 0.033	0.512 ± 0.022	0.964 ± 0.090	0.962 ± 0.086
	UNet	Pointscatter [40]	0.7787 ± 0.0142	0.8750 ± 0.0156	0.9778 ± 0.0035	0.748 ± 0.016	0.242 ± 0.033	0.620 ± 0.027	0.800 ± 0.076	0.908 ± 0.074
	UNet	TopoLoss [14]	0.7797 ± 0.0150	0.8758 ± 0.0164	0.9781 ± 0.0035	0.749 ± 0.017	0.238 ± 0.032	0.439 ± 0.018	0.780 ± 0.076	0.727 ± 0.071
	TransUNet [4]	softDice	0.7620 ± 0.0169	0.8588 ± 0.0182	0.9766 ± 0.0038	0.730 ± 0.019	0.248 ± 0.034	0.734 ± 0.027	0.933 ± 0.079	1.017 ± 0.075
	LinkNet34 [3]	softDice	0.7752 ± 0.0151	0.8747 ± 0.0161	0.9775 ± 0.0036	0.744 ± 0.017	0.243 ± 0.033	0.489 ± 0.021	0.773 ± 0.076	0.771 ± 0.072
	D-Linknet34 [48]	softDice	0.7752 ± 0.0149	0.8743 ± 0.0161	0.9775 ± 0.0036	0.744 ± 0.017	0.244 ± 0.033	0.504 ± 0.022	0.765 ± 0.075	0.777 ± 0.072
	UNet	softDice+Ours	**0.7849 ± 0.0139**	0.8816 ± 0.0151	**0.9783 ± 0.0035**	**0.754 ± 0.016**	**0.237 ± 0.033**	**0.386 ± 0.016**	**0.754 ± 0.076**	**0.672 ± 0.070**
	UNet	clDice+Ours	0.7851 ± 0.0137	**0.8844 ± 0.0148**	0.9779 ± 0.0036	0.754 ± 0.016	0.241 ± 0.033	0.393 ± 0.018	0.879 ± 0.085	0.784 ± 0.082

volumetric and distribution metrics underscore our method's effectiveness in the precision and reliability in segmenting and clustering accuracy. Most notably, the substantial advancements in topological metrics, particularly in the β_0 error, highlight our method's proficiency in capturing the intricate branch-level features of tubular structures.

Moreover, the comprehensive comparisons presented in Table 3 across various datasets underscore the superiority of our method over current state-of-the-art techniques. The results unequivocally illustrate that our approach excels in all evaluated metrics, outstripping other methodologies. The crux of this advancement lies in the exploitation of branch-level features. Compared to our approach, traditional pixel-level classification strategies, whether employing grid representations like softDice [26] or clDice [37], or point representations akin to Pointscatter [40], inherently lack in capturing the essential branch-level features. Innovatively, Our method incorporates the graph representation to capture explicit branch-level features, and morphs the predicted graphs to topologically accurate centerline masks, which are subsequently utilized for post-processing. The excellent performance of various metrics, especially topological metrics, proves the validity of our approach.

4.3 Ablation Study

Size of ROI (H). See Table 4, the experimental results on the DRIVE dataset suggest that a default ROI size of $H = 32$ provides a favorable balance across the evaluation metrics. Specifically, when H is reduced, an increase in β_0 error is observed, indicating diminished topological accuracy. Conversely, increasing H does not offer substantial metric improvements and leads to higher computational demands. Hence, $H = 32$ is adopted as the default ROI size for all three medical datasets. Similarly, the default ROI size for the MassRoad dataset has been determined to be $H = 48$.

Threshold in the Morph Module (p_{thresh}). We conduct experiments on the centerline extraction task. A smaller value of p_{thresh} denotes a more stringent selection criterion, with $p_{thresh} = 1.0$ indicating that all paths are considered without any selection filter. As detailed in Table 5, the best results were achieved at $p_{thresh} = 0.5$, which is adopted as the default setting across all experiments.

Post-processing on the Segmentation Task. As shown in Table 6, incorporating post-processing has led to a slight improvement in volumetric metrics and a significant elevation in topological metrics. Notably, the substantial enhancement in the β_0 metric primarily results from the successful suppression of false positives, which aligns with our initial hypothesis. For a qualitative demonstration, we direct the reader to the visual comparisons presented in the Appendix G.2.

https://doi.org/10.52202/079017-2187

Table 4: Effect of ROI size H on two tasks.

Dataset	H	Segmentation			Centerline Extraction	
		Dice	clDice	β_0 error	Dice	β_0 error
DRIVE	16	82.61	82.96	0.763	74.36	0.771
	32	82.44	82.78	0.692	74.97	0.555
	48	82.43	82.78	0.676	74.90	0.492
	64	82.44	82.63	0.666	74.72	0.540
MassRoad	16	78.16	87.66	0.457	61.36	2.265
	32	78.33	87.90	0.414	62.90	0.944
	48	78.53	88.16	0.386	63.59	0.620
	64	78.35	87.93	0.396	63.27	0.539

Table 5: Effect of p_{thresh}.

Dataset	p_thresh	Dice	ACC	β_0 error	β_1 error	χ error
DRIVE	0.25	70.90	97.67	0.815	1.528	1.121
	0.5	74.97	97.76	0.555	1.074	0.893
	0.75	74.97	97.71	0.572	1.221	1.025
	1.0	74.80	97.68	0.604	1.270	1.069

Table 6: Effect of post-processing.

Dataset	Method	Dice	clDice	β_0 error	β_1 error	χ error
DRIVE	softDice+Ours	**82.44**	**82.78**	**0.692**	**0.932**	**0.951**
	w.o. Post-processing	82.41	82.68	0.932	0.932	1.199
MassRoad	softDice+Ours	78.44	**87.88**	**0.374**	**0.756**	**0.655**
	w.o. Post-processing	**78.45**	87.83	0.480	0.757	0.757

4.4 Qualitative Results

We qualitatively analyze the role of GraphMorph in reducing FNs, FPs, and TEs on the segmentation and centerline extraction tasks, as detailed in Figure 4. The results show that our method can effectively reduce the three types of errors in both tasks. This is due to the utilization of branch-level features during training and the effective design of inference process. Furthermore, we visualize the graphs predicted by the Graph Decoder as well as the role of the Morph Module in suppressing various errors in Appendix G.1.

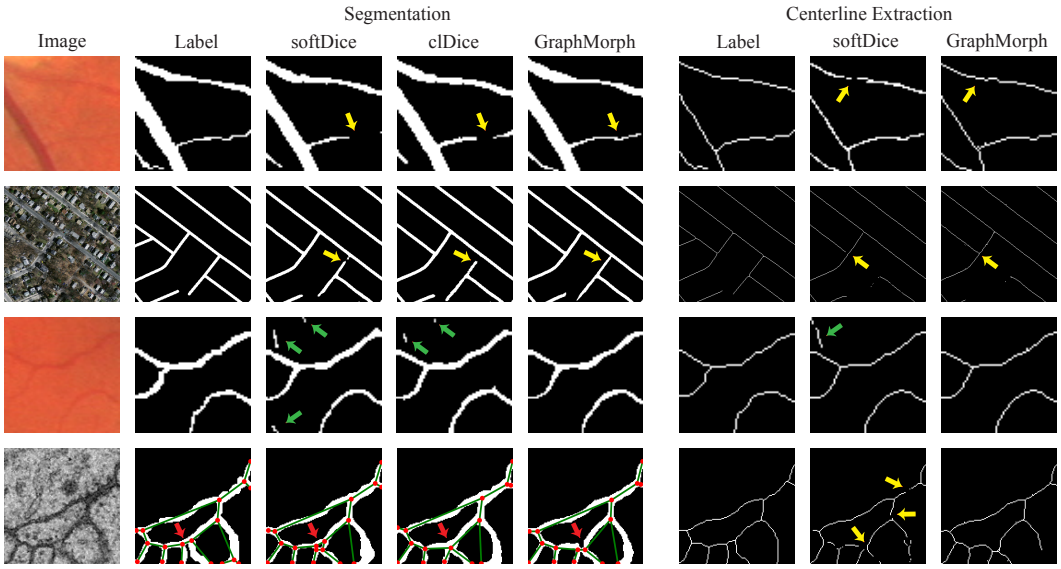


Figure 4: Visual comparison for our GraphMorph with other methods (zoom for details). Areas indicated by yellow arrows show false negatives (FNs), areas pointed by green arrows demonstrate false positives (FPs), and regions highlighted by red arrows are topological errors (TEs) identifiable in other methods but are accurately resolved by our approach.

5 Conclusion

This paper introduces GraphMorph, a framework that diverges from traditional pixel-level prediction methods in tubular structure extraction. By integrating two core components, the Graph Decoder and the Morph Module, GraphMorph adaptly captures and leverages branch-level features. Equipped with our proposed link predcition module and SkeletonDijkstra algorithm, the training and inference processes of the network are efficiently carried out. For the segmentation task, it further employs a straightforward yet effective post-processing strategy that substantially reduces false positives in the predictions. Extensive evaluations across various datasets for medical image segmentation and road network extraction have demonstrated the superiority of GraphMorph over existing methods, particularly in terms of topological metrics. This breakthrough not only boosts precision in application-specific tasks but also sets a robust foundation for future research in tubular structure extraction.

https://doi.org/10.52202/079017-2187

Acknowledgements

This work is supported by National Science and Technology Major Project (2022ZD0114902) and National Science Foundation of China (NSFC62276005).

References

[1] 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*, 9:152591, 2015.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[3] Abhishek Chaurasia and Eugenio Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE visual communications and image processing (VCIP)*, pages 1–4. IEEE, 2017.

[4] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[6] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

[7] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1959.

[8] Muhammad Moazam Fraz, Paolo Remagnino, Andreas Hoppe, Bunyarit Uyyanonvara, Alicja R Rudnicka, Christopher G Owen, and Sarah A Barman. Blood vessel segmentation methodologies in retinal images–a survey. *Computer methods and programs in biomedicine*, 108(1):407–433, 2012.

[9] Pedro Guimaraes, Jeffrey Wigdahl, and Alfredo Ruggeri. A fast and efficient technique for the automatic tracing of corneal nerves in confocal microscopy. *Translational vision science & technology*, 5(5), 2016.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[11] Songtao He, Favyen Bastani, Satvat Jagwani, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Mohamed M Elshrif, Samuel Madden, and Mohammad Amin Sadeghi. Sat2graph: Road graph extraction through graph-tensor encoding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 51–67. Springer, 2020.

[12] Congrui Hetang, Haoru Xue, Cindy Le, Tianwei Yue, Wenping Wang, and Yihui He. Segment anything model for road network graph extraction. *arXiv preprint arXiv:2403.16051*, 2024.

[13] AD Hoover, Valentina Kouznetsova, and Michael Goldbaum. Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. *IEEE Transactions on Medical imaging*, 19(3):203–210, 2000.

[14] Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. *Advances in neural information processing systems*, 32, 2019.

[15] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

 https://doi.org/10.52202/079017-2187

[16] Qiangguo Jin, Zhaopeng Meng, Tuan D Pham, Qi Chen, Leyi Wei, and Ran Su. Dunet: A deformable network for retinal vessel segmentation. *Knowledge-Based Systems*, 178:149–162, 2019.

[17] Siddhesh Khandelwal and Leonid Sigal. Iterative scene graph generation. *Advances in Neural Information Processing Systems*, 35:24295–24308, 2022.

[18] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[19] Sanjoy Kundu and Sathyanarayanan N Aakur. Is-ggt: Iterative scene graph generation with generative transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6292–6301, 2023.

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[22] Ange Lou, Shuyue Guan, and Murray Loew. Dc-unet: rethinking the u-net architecture with dual channel efficient cnn for medical image segmentation. In *Medical Imaging 2021: Image Processing*, volume 11596, pages 758–768. SPIE, 2021.

[23] Jie Mei, Rou-Jing Li, Wang Gao, and Ming-Ming Cheng. Coanet: Connectivity attention network for road extraction from satellite imagery. *IEEE Transactions on Image Processing*, 30: 8540–8552, 2021.

[24] Marina Meilă. Comparing clusterings—an information based distance. *Journal of multivariate analysis*, 98(5):873–895, 2007.

[25] Martin J Menten, Johannes C Paetzold, Veronika A Zimmer, Suprosanna Shit, Ivan Ezhov, Robbie Holland, Monika Probst, Julia A Schnabel, and Daniel Rueckert. A skeletonization algorithm for gradient-based optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21394–21403, 2023.

[26] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016.

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

[28] Volodymyr Mnih and Geoffrey E Hinton. Learning to detect roads in high-resolution aerial images. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part VI 11*, pages 210–223. Springer, 2010.

[29] Agata Mosinska, Pablo Marquez-Neila, Mateusz Koziński, and Pascal Fua. Beyond the pixel-wise loss for topology-aware delineation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3136–3145, 2018.

[30] Lei Mou, Yitian Zhao, Li Chen, Jun Cheng, Zaiwang Gu, Huaying Hao, Hong Qi, Yalin Zheng, Alejandro Frangi, and Jiang Liu. Cs-net: Channel and spatial attention network for curvilinear structure segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part I 22*, pages 721–730. Springer, 2019.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[32] Chinmay Prabhakar, Suprosanna Shit, Johannes C Paetzold, Ivan Ezhov, Rajat Koner, Hongwei Li, Florian Sebastian Kofler, and Bjoern Menze. Vesselformer: Towards complete 3d vessel graph generation from images. In *Medical Imaging with Deep Learning*, pages 320–331. PMLR, 2024.

[33] Yaolei Qi, Yuting He, Xiaoming Qi, Yuan Zhang, and Guanyu Yang. Dynamic snake convolution based on topological geometric constraints for tubular structure segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6070–6079, 2023.

[34] 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: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[35] Mojtaba Seyedhosseini, Mehdi Sajjadi, and Tolga Tasdizen. Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2168–2175, 2013.

[36] Seung Yeon Shin, Soochahn Lee, Il Dong Yun, and Kyoung Mu Lee. Deep vessel segmentation by learning graphical connectivity. *Medical image analysis*, 58:101556, 2019.

[37] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16560–16569, 2021.

[38] Suprosanna Shit, Rajat Koner, Bastian Wittmann, Johannes Paetzold, Ivan Ezhov, Hongwei Li, Jiazhen Pan, Sahand Sharifzadeh, Georgios Kaissis, Volker Tresp, et al. Relationformer: A unified framework for image-to-graph generation. In *European Conference on Computer Vision*, pages 422–439. Springer, 2022.

[39] 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*, 23(4):501–509, 2004.

[40] Dong Wang, Zhao Zhang, Ziwei Zhao, Yuhang Liu, Yihong Chen, and Liwei Wang. Pointscatter: Point set representation for tubular structure extraction. In *European Conference on Computer Vision*, pages 366–383. Springer, 2022.

[41] Yan Wang, Xu Wei, Fengze Liu, Jieneng Chen, Yuyin Zhou, Wei Shen, Elliot K Fishman, and Alan L Yuille. Deep distance transform for tubular structure segmentation in ct scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3833–3842, 2020.

[42] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[43] Zhenhua Xu, Yuxuan Liu, Lu Gan, Yuxiang Sun, Xinyu Wu, Ming Liu, and Lujia Wang. Rngdet: Road network graph detection by transformer in aerial images. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–12, 2022.

[44] Zhenhua Xu, Yuxuan Liu, Yuxiang Sun, Ming Liu, and Lujia Wang. Rngdet++: Road network graph detection by transformer with instance segmentation and multi-scale features enhancement. *IEEE Robotics and Automation Letters*, 2023.

[45] Ziyun Yang and Sina Farsiu. Directional connectivity-based segmentation of medical images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11525–11535, 2023.

[46] Tongjie Y Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.

[47] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.

 https://doi.org/10.52202/079017-2187

[48] Lichen Zhou, Chuang Zhang, and Ming Wu. D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 182–186, 2018.

[49] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[50] Kelly H Zou, Simon K Warfield, Aditya Bharatha, Clare MC Tempany, Michael R Kaus, Steven J Haker, William M Wells III, Ferenc A Jolesz, and Ron Kikinis. Statistical validation of image segmentation quality based on a spatial overlap index1: scientific reports. *Academic radiology*, 11(2):178–189, 2004.

Appendix

A Implementation Details

In this section, we provide more implementation details of GraphMorph.

For fair comparison to previous works like PointScatter [40], we use the ADAM optimizer with the initial learning rate 1e-3 and cosine learning rate schedule with warm-up strategy to train the network. The weight decay is set to be 1e-4 uniformly. We train the network for 3K iterations for the three medical image datasets, and 10K for MassRoad. We use batchsize=4 for all datasets. We implement GraphMorph based on PyTorch [31] and Detectron2 [42].

Our $\mathrm{SkeletonDijkstra}$ algorithm is designed to run solely on CPU due to its computational nature.We use $p_{thresh} = 0.5$ across all experiments. To enhance performance and efficiency, we have implemented this algorithm in C++. For a detailed understanding of the algorithm, refer to the pseudo-code provided in Appendix D.

B Graph Construction

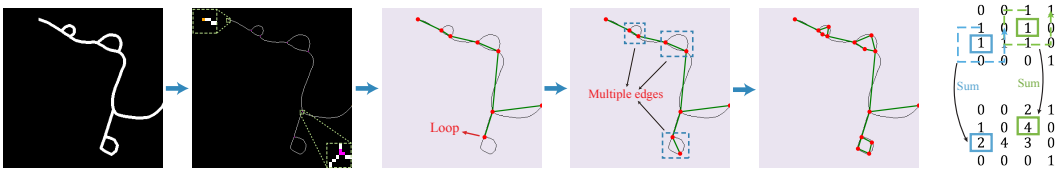
The process of constructing a graph can be briefly summarized in three steps as follows: (1) Generate the centerline mask of the tubular structure using skeletonization algorithm [46]; (2) Analyze each centerline point P by counting the centerline points among its eight neighbors (denoted as N, not including P). Define P as a junction if $N \geq 3$ and as an endpoint if $N = 1$. Points with $N = 2$ are path points and are not considered as nodes. Junctions and endpoints form the node set V of the graph G. (3) If there is a pathway consisting of only path points between two nodes, then there is an edge between them. All edges form the edge set E. We use publicly accessible implementations of skeletonization[1] and graph construction.[2]

However, the resultant graph may contain elements such as *loops* (closed paths where a node connects back to itself) and *multiple edges* (more than one edge connecting the same pair of nodes). Addressing both these elements is crucial; otherwise, reconstructing such structures during the inference process would be challenging. As depicted in Figure 5, to manage these complexities, we introduce new nodes in the following manner:

1. *Loops*: We insert new nodes at selected points within the loop to break the cycle.

2. *Multiple edges*: After resolving loops, we then add nodes along edges where multiple connections exist between the same pair of nodes.

These modifications ensure that the graph structure is simplified and ready for more effective processing during inference.



(a) (b)

Figure 5: (a) Stages of graph construction from the binary mask of a road network. The first stage demonstrates skeletonization process to a centerline mask. In the second image, we highlight the endpoints in orange and junctions in purple. Adjacent junctions are merged and considered as a single junction. Subsequent stages illustrate resolving Loops and reducing Multiple edges. (b) Example of calculating N.

[1] skimage.morphology.skeletonize
[2] https://github.com/Image-Py/sknw

https://doi.org/10.52202/079017-2187

Table 7: Comparison of [rln]-token and our dynamic module on the DRIVE and STARE datasets. The "Time" metric represents the cumulative time required to process all sliding windows in the link prediction phase for a single 384×384 image patch during inference.

Dataset	Method	Time/s	Node Detection (↑)		Edge Detection (↑)		Volumetric metrics (↑)		Topological metrics (↓)		
			AP@0.5	AR@0.5	AP@0.5	AR@0.5	Dice	ACC	β_0 error	β_1 error	χ error
DRIVE	[rln]-token	0.2823	52.40	58.20	23.33	37.78	74.95	97.76	0.548	1.026	0.866
	Dynamic	0.1582	52.30	58.11	23.25	37.89	74.97	97.76	0.555	1.074	0.893
STARE	[rln]-token	0.1385	54.95	61.44	27.84	43.73	73.99	98.92	0.483	0.731	0.663
	Dynamic	0.0754	55.06	61.90	27.80	43.62	74.25	98.94	0.482	0.799	0.653

C Link Prediction Module

To validate the effectiveness of our dynamic link prediction module, we compare it with the approach used in RelationFormer [38].

RelationFormer learns an additional [rln]-token during the training of DETR to encode the relationships between node queries. In the inference stage, to predict the connection between two nodes, the method concatenates the features of the two nodes with the [rln]-token into a single vector. This vector is then processed by a three-layer MLP to predict the probability of connection between the nodes. Let us assume that there are P matched queries, denoted as $\widetilde{Q}^r \in \mathbb{R}^{P\times C}$. In RelationFormer, the connection probability for each node pair is computed individually, resulting in a computational complexity of $O(P^2 \times C^2)$. In contrast, our dynamic module achieves a complexity of $O(P \times C^2)$ (Equation (4) to (6)), reducing the computational burden.

Table 7 presents a comparison between the [rln]-token and our dynamic module on the task of centerline extraction. The metrics "Node Detection" and "Edge Detection" are reproduced from RelationFormer, which measure the accuracy of the nodes and edges extracted by the Graph Decoder. Other metrics assess the accuracy of the centerline masks output by the Morph Module. All experiments are based on UNet. The results demonstrate that both methods achieve comparable performance. However, our method significantly reduces the computational complexity, thereby shortening the inference time. This enhancement makes our approach more suitable for applications requiring fast processing speeds without sacrificing performance.

D SkeletonDijkstra Algorithm

The pseudo-code of our $\mathrm{SkeletonDijkstra}$ algorithm is given in Algorithm 2, which finds the optimal path satisfying the skeleton nature for two points.

E Details of Processing in Segmentation

E.1 Soft Skeletonization

See $\mathrm{soft\_skeleton}$ function in Listing 1. In the segmentation task, the segmentation network outputs the segmentation probability S_m, which we need to soft skeletonized into the centerline probability P_m for input into the Morph Module.

E.2 Post-processing to Suppress False Postives

See $\mathrm{dilate\_with\_seg\_limit}$ function in Listing 1. In the segmentation task, after obtaining topologically accurate centerline masks by the Morph Module, false positives can be greatly suppressed with this post-processing strategy.

F Computational Resources

Hardware Configuration. Experiments were conducted using an NVIDIA GeForce RTX 3090 with 24 GB GPU memory. The CPU used was an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz, which features 28 cores.

Analysis of training process. Training on the DRIVE dataset with a UNet backbone and a batch size of 4 using "SoftDice+Ours" method requires approximately 11.8 GB of GPU memory, compared

Algorithm 2 SkeletonDijkstra Algorithm

Input: Start point s, End point e, Cost map C, Path threshold p_{thresh}
Output: Minimum cost path from s to e under threshold p_{thresh}

```
Initialize priority queue Q with (0, [s])
Initialize visited set Vis
while not Q empty do
    (cost, path) ← Q.pop()
    curr ← last element of path
    Add curr to Vis
    if curr = e then
        avg ← cost/ length(path)
        if avg > p_thresh then
            return ∅
        end if
        return path
    end if
    for each n in neighbors of curr do
        if n in Vis then
            continue
        end if
        neis_in_path ← count of n's neighbors in path
        if neis_in_path ≤ 1 then
            path ← path concatenated with [n]
            cost ← cost + C[n.x][n.y]
            Q.push((cost, path))
        end if
    end for
end while
```

to 5.4 GB of "SoftDice". More comparison between these two methods are show in Table 8. The increase in parameters and FLOPs in our approach primarily stems from the integration of the Graph Decoder featuring a DETR module. This component is crucial for predicting accurate topological structures of the graphs. Advancements in transformer architectures that reduce computational overhead could potentially enhance the efficiency of our model during training.

Inference time analysis. The inference times for each model component are summarized in Table 9. The data presented in the table was obtained by processing a 384 $\times$ 384 image patch from the DRIVE dataset. ROI size is $H = 32$, with a sliding window stride of 30. As shown in the table, significant time is concentrated on the Morph Module. The primary time expenditure currently arises from processing the patches sequentially in our sliding window strategy. However, as each patch operates independently, there is significant potential to enhance efficiency by parallelizing the computations of all patches. Recognizing this opportunity, we plan to focus future work on optimizing the Morph Module by implementing parallel processing techniques to accelerate inference.

Table 8: Comparison of required resources during training.

Method	Params	FLOPs	Time per iteration (s)	GPU Memory
SoftDice	39M	187G	0.203	5.4 GB
softDice+Ours	48M	268G	0.589	11.8 GB

Table 9: Inference timing for each Module.

Module	Device	Time (s)
Segmentation network	GPU	0.0101
Deformable DETR	GPU	0.1194
Link prediction module	GPU	0.0379
Morph Module	CPU	0.2933
(Segmentation) Post-processing	CPU	0.0279

https://doi.org/10.52202/079017-2187

Listing 1 Python codes of the soft skeletonization operation and post-processing strategy during the inference process of the segmentation task.

```python
import numpy as np
import cv2
from skimage.morphology import skeletonize
from scipy.ndimage import distance_transform_edt

def soft_skeleton(seg_prob: np.ndarray):
    '''
    Generate centerline probability map from the segmentation probability map.
    seg_prob: segmentation probability map output from the segmentation network.
    '''
    seg_mask = seg_prob > 0.5
    ske = skeletonize(seg_mask)
    distmap = distance_transform_edt(~ske.astype(np.bool_))
    distmap = np.clip(distmap, 0, 64) / 64
    cline_prob = seg_prob * (1 - distmap)
    return cline_prob

def dilate_with_seg_limit(cline_mask, seg_mask, kernel_size=3):
    '''
    Post process in the segmentation task to suppress false postives.
    cline_mask: centerline mask output from the Morph module.
    seg_mask: segmentation mask thresholded from the segmentation probability map.
    kernel_size: kernel size of the dilation operation.
    '''
    kernel = np.ones((kernel_size, kernel_size), np.uint8)
    res = np.minimum(cline_mask, seg_mask)
    while True:
        dilated_cline_mask = cv2.dilate(res, kernel, iterations=1)
        dilated_res = dilated_cline_mask - res
        dilated_res = np.minimum(dilated_res, seg_mask)
        dilated_cline_mask = res + dilated_res
        if np.array_equal(dilated_cline_mask, res):
            break
        res = dilated_cline_mask
    return dilated_cline_mask
```

G More Visualization Results

G.1 Visualization of Predicted Graphs

We visualize the graphs predicted by the Graph Decoder within the context of the centerline extraction task and analyze the role of the Morph Module. As shown in Figure 6, the first four rows illustrate the Graph Decoder's robust capability to predict graphs. By comparing the final predicted results obtained through the Morph Module (last column) with those obtained by thresholding P_m at 0.5 (fourth column), it is evident that issues such as redundant and broken branches are effectively mitigated. However, the Morph Module also has limitations. Notably, the setting of p_{thresh} might lead to overlooking some true-positive edges due to inaccuracies in P_m, which is illustrating in the last two rows in Figure 6. This highlights areas for future improvement.

G.2 Visualization of Effect of Post-processing on the Segmentation Task

Figure 7 showcases the impact of post-processing in mitigating false positives within segmentation tasks, a procedure fully elaborated in Appendix E.2. This figure clearly reveals the elimination of isolated regions, originally predicted by the segmentation networks, across all datasets. Notably, the excision of such regions—often minute in scale—exerts a nominal effect on volumetric metrics while markedly bolstering topological metrics. This enhancement in the integrity of topological metrics through post-processing is substantiated by the data in Table 6.

H Limitations

Despite the advancements offered by GraphMorph, the method exhibits certain limitations. First, the reliance on post-processing for the segmentation task indicates a potential underutilization of branch-level features. Although false postives are significantly suppressed, segmentation results are not always topologically aligned with predicted graphs, suggesting room for improvement in

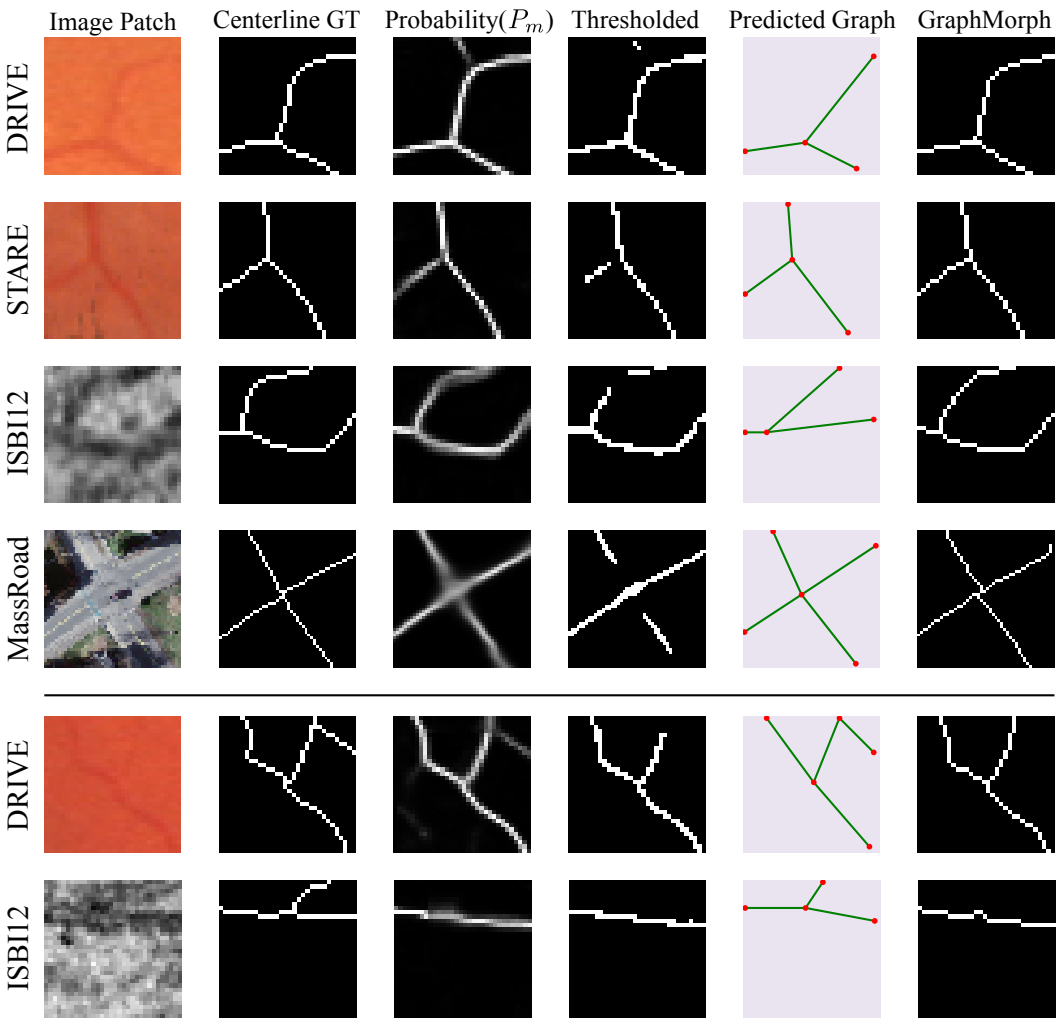


Figure 6: Visualization of intermediate results in the centerline extraction task. The results in the fourth column are obtained by thresholding P_m at 0.5. Comparisons across the first four rows illustrate that GraphMorph achieves improved results through morphing predicted graphs. The last two rows demonstrate how the settings of p_{thresh} in the Morph Module may lead to concessions to P_m, resulting in false negatives.

segmentation performance. Additionally, the necessity to train on relatively small ROIs, due to the complex nature of tubular structures, requires sliding window technique during inference. This technique may not fully capture comprehensive branch-level details and the global context of the entire tubular structure. Based on the limitations, future developments will aim to refine segmentation algorithms to utilize predicted graphs directly, thereby reducing dependency on post-processing. Concurrently, efforts will also focus on the capability of processing larger fields of view in a single analysis, thus preserving global context and enhancing feature consistency across the entire structure.

I Additional Experiments on 3D Dataset

The application of GraphMorph to 3D medical datasets can be initially explored for its clinical significance. Thus, we have extended GraphMorph to use the 3D UNet architecture and tested it on the pulmonary arterial vascular segmentation dataset from the PARSE challenge, which includes 100 annotated 3D CT scans. These cases were divided in a 7:1:2 ratio for training, validation, and testing.

The preliminary results, as detailed in Table 10, show that our method consistently outperforms existing baselines across all metrics, mirroring the success we observed with 2D data. This alignment

https://doi.org/10.52202/079017-2187

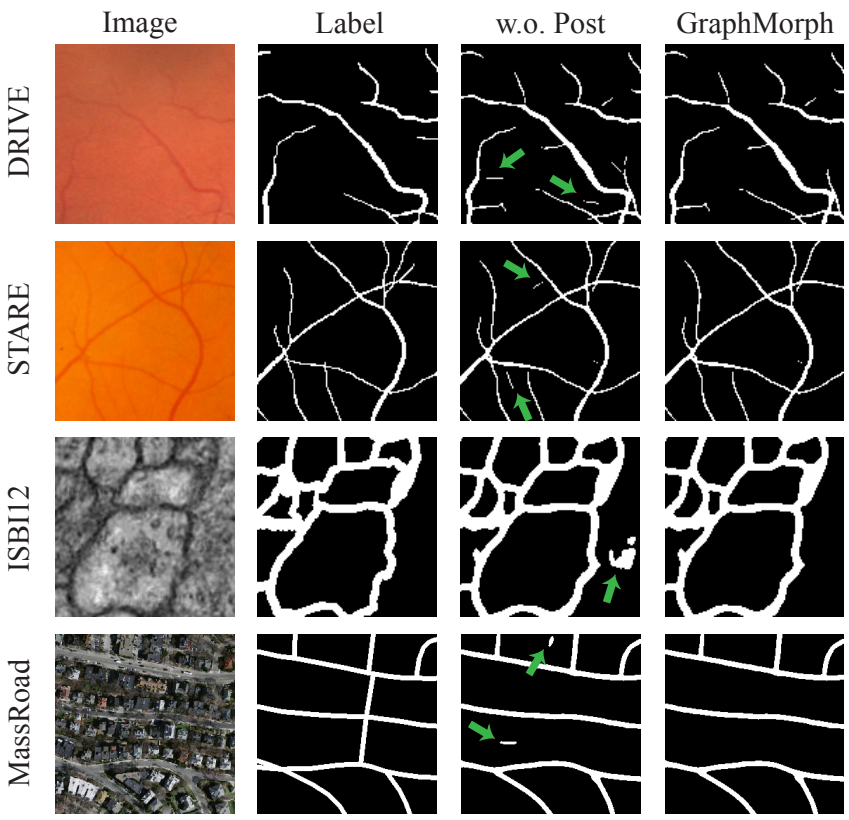


Figure 7: Visualization of the effect of the post-processing in the segmentation task across four datasets. Columns represent, from left to right: original images, ground truth segmentation labels, thresholded output from the segmentation network (w.o. Post), and results with post-processing. Green arrows highlight areas where false positives have been successfully suppressed.

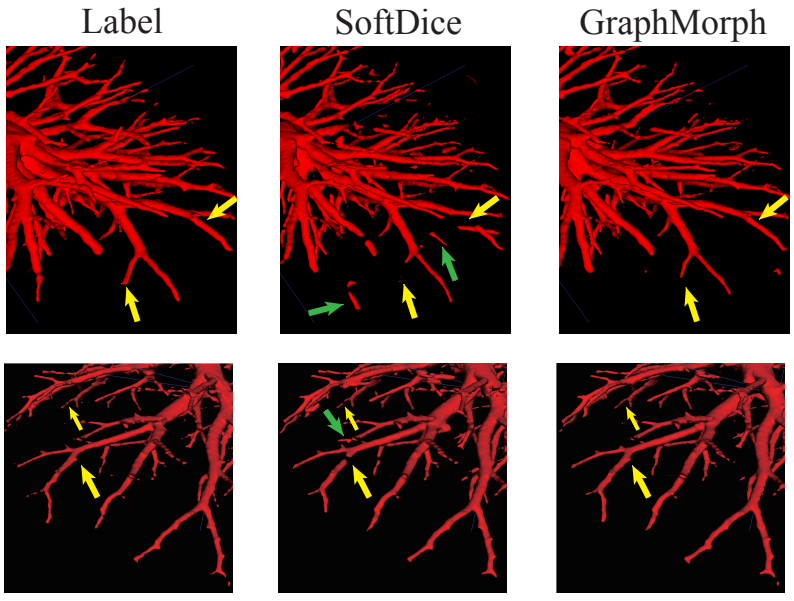


Figure 8: Visual comparison for our GraphMorph with baseline on the segmentation task. Areas indicated by yellow arrows show false negatives (FNs) and areas pointed by green arrows demonstrate false positives (FPs) appear in baseline but are accurately predicted by our approach.

https://doi.org/10.52202/079017-2187

between 2D and 3D results not only underlines the effectiveness of our method but also its adaptability to 3D vessel segmentation task, which indicates the potential of GraphMorph in clinical diagnosis. Moreover, Figure I demonstrates that GraphMorph effectively suppresses false positives (FPs) and false negatives (FNs) in fine structures. Further attempts on 3D datasets will be made to validate its effectiveness.

Table 10: Segmentation performance of GraphMorph on PARSE dataset.

Backbone	Method	Volumetric metrics (↑)			Distribution metrics		Topological metrics (↓)	
		Dice	clDice	ACC	ARI(↑)	VOI(↓)	β_0 error	χ error
UNet	softDice	0.7968 ± 0.0166	0.8350 ± 0.0152	0.9878 ± 0.0021	0.779 ± 0.017	0.134 ± 0.019	1.087 ± 0.078	1.130 ± 0.082
	softDice+Ours	**0.8196 ± 0.0138**	**0.8730 ± 0.0111**	**0.9901 ± 0.0015**	**0.805 ± 0.014**	**0.115 ± 0.015**	**0.536 ± 0.039**	**0.602 ± 0.045**

J Broader Impacts

In this work, we present GraphMorph, a framework aimed at improving the extraction of tubular structures in medical image analysis, such as blood vessels and other elongated anatomical features. Fine-scale structures often consist of interconnected branches forming cohesive networks critical to physiological functions. By enhancing topological accuracy, GraphMorph provides more coherent and precise representations of these structures. These improved predictions with better topology in medical diagnostic scenarios related to tubular structures may assist clinical diagnosis. While our results are promising, they are based on publicly available datasets that may not fully capture the complexity and variability of real-world clinical data; therefore, further validation on more 3D datasets is necessary to confirm its applicability in clinical settings. At the present stage, we do not foresee any potential negative societal impacts arising from our work. Our goal is to contribute a useful tool for the medical imaging community, supporting efforts to improve segmentation accuracy and ultimately aiding in better healthcare outcomes.

https://doi.org/10.52202/079017-2187

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The last sentence in **Abstract**; the last two paragraphs of **Introduction**.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix H.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The validity of the method is demonstrated primarily through experiments.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide a detailed description of the model architecture in Section 3, including figures, textual descriptions, and algorithmic flows. Hyperparameters are mainly provided in the "Implementation Details" paragraph of Section 4.1. There are also some details although described in detail in the main text, we provide the algorithmic flow and codes in Appendix D to ensure the reproducibility of the methodology.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

https://doi.org/10.52202/079017-2187

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: However, we provide training and infernece details exhaustively in Section 3 and 4.1. Additionally, codes of processes in segmentation are supplied in Appendix E. The information we have provided is sufficient to reproduce our work. The complete code will be made publicly available soon.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Section 4.1 and Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: The experimental setup in this study involved extensive testing across multiple datasets and with various types of backbone networks, consistently demonstrating improvements through our method. This extensive validation across diverse conditions ensures the reproducibility and reliability of the results. Additionally, the sheer volume of experiments conducted makes the computation of error bars highly time-consuming and computationally expensive.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work does not violate the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix J analyzes the positive impacts. We observe that there are no significant negative effects of the methodology.

https://doi.org/10.52202/079017-2187

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper is proposing an AI algorithm for medical assistance that does not run such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The datasets used in the paper have been cited in their original literature. For the utilization of publicly available code the source is also indicated.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The code and model are not publicly available at this time. We will make them available later.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

https://doi.org/10.52202/079017-2187

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.