Preservation of Topology using Deep Image segmentation

Debayan Datta

July, 2024

Abstract

Segmentation algorithms frequently encounter topological errors, particularly in handling fine-scale structures such as fragmented connections. A novel method is proposed that addresses this challenge by introducing a continuous-valued loss function. This function is designed to enforce segmentations to maintain correct topology akin to the ground truth, ensuring consistent Betti numbers. The innovative topology-preserving loss function is differentiable, allowing seamless integration into end-to-end training of deep neural networks.

Introduction

Image segmentation, which involves assigning labels to every pixel in an input image, is vital for numerous computer vision applications. Advanced deep segmentation techniques achieve high per-pixel accuracy by learning robust feature representations through end-to-end deep network training. Despite this, these methods often falter on fine-scale structures, such as small object instances, objects with several connected components, and thin connections. These fine-scale structures are essential for understanding the functionality of the objects.

For instance, precise extraction of delicate features like wires and cables is critical in electrical engineering tasks, such as circuit design or troubleshooting. In medical imaging, accurate delineation of thin structures like blood vessels and nerve fibers is crucial for providing reliable morphological and structural analysis of the biological system. A broken connection or a missing component might result in a minor per-pixel error but can lead to significant functional inaccuracies.

Proposed Solution

- Introduce a continuous, differentiable loss function that ensures the segmented image preserves the same topology as the ground truth by maintaining the same Betti numbers (number of connected components and handles).
- A neural network trained with this loss function achieves high topological fidelity and per-pixel accuracy.
- However Betti numbers are discrete values but a continuous, differentiable measure of topological similarity between a prediction and ground truth is needed for back-propagation through the network.
- We require a continuous function that summarizes the topological information. Hence we use the likelihood function f that is predicted by the neural network.

Methodology

The topological similarity between the likelihood f and the ground truth is called the topological loss. The topological loss is tends to zero, the segmentation is guaranteed to be topologically correct, i.e., have identical topology as the ground truth. It achieves superior performance on metrics that promote structural accuracy. Specifically, our method gives good results in terms of Betti number error, which precisely measures topological accuracy.

- We focus on 0 and 1-dimensional topology on 2-dimensional images therefore we consider mostly on bridges and connected components in the image domain.
- Develop a deep neural network with a new topological loss, $L_{\text{topo}}(f, g)$, where f is the predicted likelihood map and g is the ground truth.
- The total loss function combines the per-pixel cross-entropy loss, $L_{\text{bce}}(f, g)$, and the topological loss, weighted by λ:

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

  λ controls the weight of the topological loss and the likelihood function f, whose value ranges from 0 to 1.
- Our method shows how topological computation and deep learning can be mutually beneficial. We incorporate the loss in different models and try to check the accuracy of each ones.

Persistent Homology

- Given a continuous image domain, $\Omega \subseteq \mathbb{R}^2$, we study a likelihood map $f(x) : \Omega \to \mathbb{R}$, which is predicted by a deep neural network.
- A segmentation $X \subseteq \Omega$ is calculated by thresholding f at a given value, often at 0.5.
- Simply comparing Betti numbers of X and g will result in a discrete-valued topological error function. To incorporate topological priors into deep neural networks, we need a continuous-valued function that can reveal subtle differences between similar structures; hence, we choose f.
- For a continuous-valued function f, its persistence diagram, $\text{Dgm}(f)$, contains a finite number of dots in a 2-dimensional plane, called persistent dots. Each coordinate of a persistent dot $p \in \text{Dgm}(f)$ corresponds to the birth and death of the topological structure.

Cubical Discretization

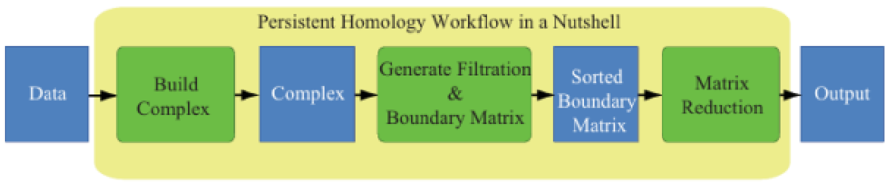
- We first discretize an image patch into vertices (pixels), edges, and squares, using a cubical complex discretization suitable for images.
- We take the help of the work by [6] which mainly focused on handling data of large size and high dimension, and therefore, makes the persistence computation of cubical data more feasible.
- We focus on uniformly/regularly sampled data which is common in visualization and data analysis, i.e. image data consisting of pixels or voxels. We use the name 'cubical' for such data.
- **Cubical Complexes:** An elementary interval is defined as a unit interval $[k, k+1]$. For a d-dimensional space, a cube is a product of d elementary intervals. A cubical complex of dimension d is a collection of cubes of dimension at most d. Similarly to the definition of a simplicial complex, it must be closed under taking faces and intersections.
- **Boundary Matrix:** For any d-dimensional cell (simplex or cube), its boundary consists of its $(d-1)$-dimensional faces. The boundary of a set of cells is the modulo 2 sum of the boundaries of its elements. The d-dimensional boundary operator is an $n_{d-1} \times n_d$ binary matrix, with columns representing the boundaries of d-cells and rows representing $(d-1)$-cells.
- Cohen-Steiner proved that for any two filtering functions f and g, the difference of their persistence is always upper-bounded by the L_∞ norm of their difference:

  $$\|f - g\|_\infty = \max_{x \in X} |f(x) - g(x)|$$

  This guarantees that persistence can be used as a signature. Whenever two persistence outputs are different, we know that the functions are definitely different.

Persistence Computation

- [2] devised an algorithm to compute persistent homology, which works in cubic time (in the size of a complex).
- In case of images, function f is defined on all pixels/voxels. These values are interpreted as values of vertices of a complex.
- The filtration of the complex is computed and the sorted boundary matrix is generated. This matrix is then reduced by a modified Gausian Elimination Method. And from here we get the Persistence Diagram.
- **Filtration Building Algorithm:** Function is extended to all cells of the complex, by assigning each cell the maximum value of its vertices. Then, all cells are sorted in ascending order according to the function value, so that each cell is added to the filtration after all of its faces. Such a sequence of cells is called a **lower-star filtration.**
- Reduction Algorithm: In the reduction step, the algorithm performs column reductions on the sorted boundary matrix from left to right. Each new column is reduced by addition with the already reduced columns, until its lowest nonzero entry is as high as possible. The reduced matrix encodes all the persistent homology information.


Persistent Homology Workflow in a Nutshell
Data
Build Complex
Complex
Generate Filtration & Boundary Matrix
Sorted Boundary Matrix
Matrix Reduction
Output

A workflow of the persistent homology computation

Topological Loss and Gradient

- Use persistence diagrams to capture all possible topological structures across different thresholds by modifying the Wasserstein distance to measure topological similarity between the predicted and ground truth persistence diagrams.
- For persistence diagrams,Dgm(f) and Dgm(g), we find a best one-to-one correspondence between these sets, and measure the total squared distance between them. An unmatched dot will be matched to the diagonal line.
- Let Γ be the set of all possible bijections between Dgm(f) and Dgm(g).
The loss $L_{\text{topo}}(f, g)$ is:

$$L_{\text{topo}}(f,g) = \min_{\gamma \in \Gamma} \sum_{p \in \text{Dgm}(f)} \|p - \gamma(p)\|_2^2$$

$$= \sum_{p \in \text{Dgm}(f)} \left([\text{birth}(p) - \text{birth}(\gamma^*(p))]^2 + [\text{death}(p) - \text{death}(\gamma^*(p))]^2 \right)$$

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

- This loss measures the minimal amount of necessary effort to modify the diagram of Dgm(f) to Dgm(g) by moving all dots toward their matches.

Theorem: When $L_{\text{topo}}(f, g) = 0$, the segmentation thresholded f at 0.5 has the same Betti numbers as the ground truth. Therefore segmentation has same topology as that of ground truth.

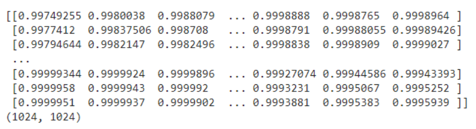
Topological Gradient

- The loss function depends on crucial thresholds where topological changes occur, such as the birth and death times of dots in the diagram. These thresholds are determined critical points with zero gradients in a differentiable function f. For such functions, critical points always correspond to pixels, as topological changes happen at these points.
- $L_{\text{topo}}(f, g)$ is a piecewise differentiable loss function over the space of all possible likelihood functions f. We find the gradient of $L_{\text{topo}}(f, g)$.
- During training, we take the negative gradient direction of $L_{\text{topo}}(f, g)$.
- For each topological structure, the gradient descent step is pushing the corresponding dot $p \in \text{Dgm}(f)$ toward its match $\gamma^*(p) \in \text{Dgm}(g)$.
- We denote by $c_b(p)$ and $c_d(p)$ the birth and death critical points of the corresponding topological structure. The gradient of the topological loss $\nabla_\omega L_{\text{topo}}(f, g)$ is:

$$\sum_{p \in \text{Dgm}(f)} 2\left[f(c_b(p)) - \text{birth}(\gamma^*(p))\right] \frac{\partial f(c_b(p))}{\partial \omega} + 2\left[f(c_d(p)) - \text{death}(\gamma^*(p))\right] \frac{\partial f(c_d(p))}{\partial \omega}$$

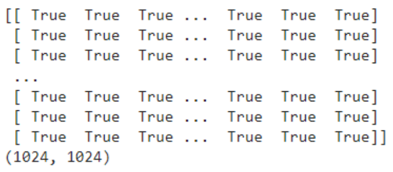
Persistence Homology Implementation

- For example we take a normalized array of size (1024×1024) and referred to it as the Likelihood. We then calculated the ground truth from this likelihood matrix.



```
[[0.99749255 0.9980038  0.9988079  ... 0.9998888  0.9998765  0.9998964 ]
 [0.9977412  0.99837506 0.998708   ... 0.9998791  0.99988055 0.99989426]
 [0.99794644 0.9982147  0.9982496  ... 0.9998838  0.9998909  0.9999027 ]
 ...
 [0.99999344 0.9999924  0.9999896  ... 0.99927074 0.99944586 0.99943393]
 [0.9999958  0.9999943  0.999992   ... 0.9993231  0.9995067  0.9995252 ]
 [0.9999951  0.9999937  0.9999902  ... 0.9993881  0.9995383  0.9995939 ]]
(1024, 1024)
```

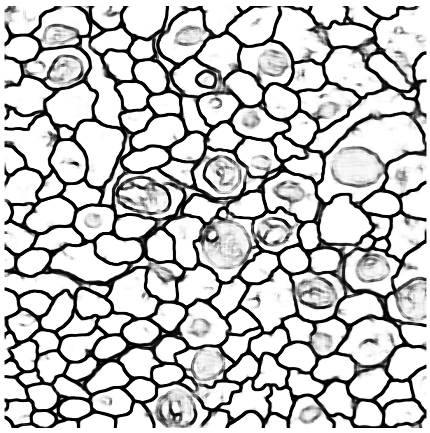
Likelihood Matrix with shape



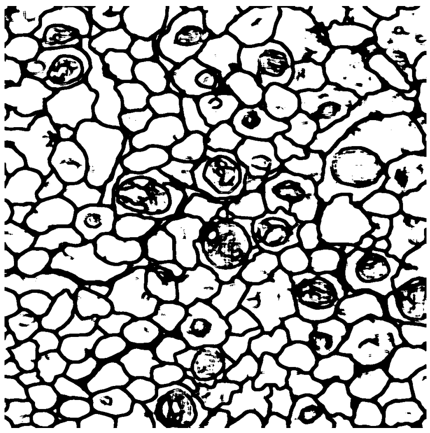
```
[[ True  True  True ...  True  True  True]
 [ True  True  True ...  True  True  True]
 [ True  True  True ...  True  True  True]
 ...
 [ True  True  True ...  True  True  True]
 [ True  True  True ...  True  True  True]
 [ True  True  True ...  True  True  True]]
(1024, 1024)
```

Groundtruth Matrix with shape

- The groundtruth array is saved as an image. This is how both the likelihood and groundtruth looks.

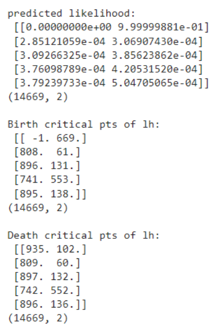


Likelihood

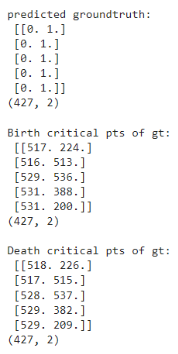


Ground Truth

- The persistence of both the Likelihood and the Ground Truth is calculated. From this, we obtain the Predicted Likelihood and Predicted Ground Truth as well as the birth and death critical points for both the Likelihood and the Ground Truth. The below images show some of the points with their shape.

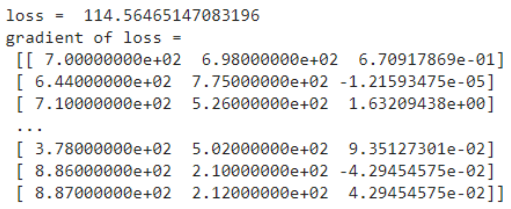


```
predicted likelihood:
 [[0.00000000e+00 9.99999881e-01]
 [2.85121059e-04 3.06907430e-04]
 [3.09266325e-04 3.85623862e-04]
 [3.76098789e-04 4.20531520e-04]
 [3.79239733e-04 5.04705065e-04]]
(14669, 2)

Birth critical pts of lh:
 [[ -1. 669.]
 [808.  61.]
 [896. 131.]
 [741. 553.]
 [895. 138.]]
(14669, 2)

Death critical pts of lh:
 [[935. 102.]
 [809.  60.]
 [897. 132.]
 [742. 552.]
 [896. 136.]]
(14669, 2)
```



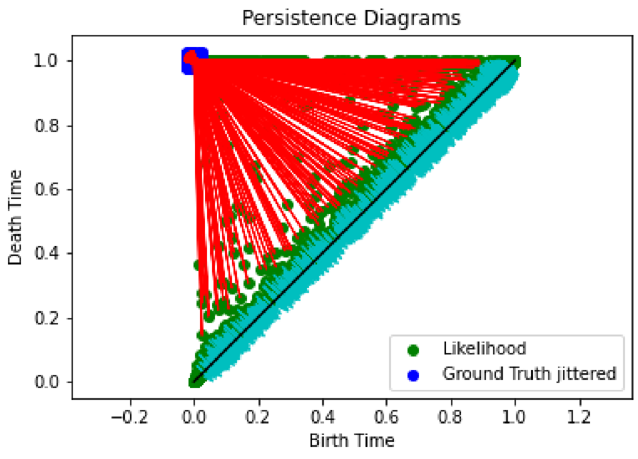
```
predicted groundtruth:
 [[0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]]
(427, 2)

Birth critical pts of gt:
 [[517. 224.]
 [516. 513.]
 [529. 536.]
 [531. 388.]
 [531. 200.]]
(427, 2)

Death critical pts of gt:
 [[518. 226.]
 [517. 515.]
 [528. 537.]
 [529. 382.]
 [529. 209.]]
(427, 2)
```

- The Total Loss and Gradient of the Loss are calculated for the entire image. This loss is entirely the Topological loss that calculates the similarity between the likelihood and the groundtruth.

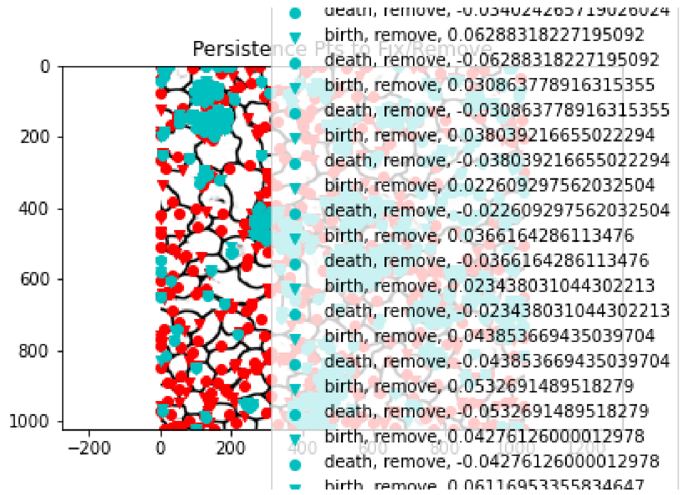


```
loss =  114.56465147083196
gradient of loss =
 [[ 7.00000000e+02  6.98000000e+02  6.70917869e-01]
 [ 6.44000000e+02  7.75000000e+02 -1.21593475e-05]
 [ 7.10000000e+02  5.26000000e+02  1.63209438e+00]
 ...
 [ 3.78000000e+02  5.02000000e+02  9.35127301e-02]
 [ 8.86000000e+02  2.10000000e+02 -4.29454575e-02]
 [ 8.87000000e+02  2.12000000e+02  4.29454575e-02]]
```

- The Persistence Diagram is shown below. The coordinates of each dot p in the diagram are $(1 - \text{birth}(p), 1 - \text{death}(p))$. The lines indicate which points need to be removed and which need to be fixed. Points that need fixing are connected to $(0, 1)$, signifying that these features appear at the very beginning and persist until the end of the persistence homology. Points that need to be removed are projected onto the diagonal. Generally, the further a point is from the diagonal in the Persistence Diagram, the more significant it is.



- On the Likelihood image, we mark the territories that need to be fixed or removed.

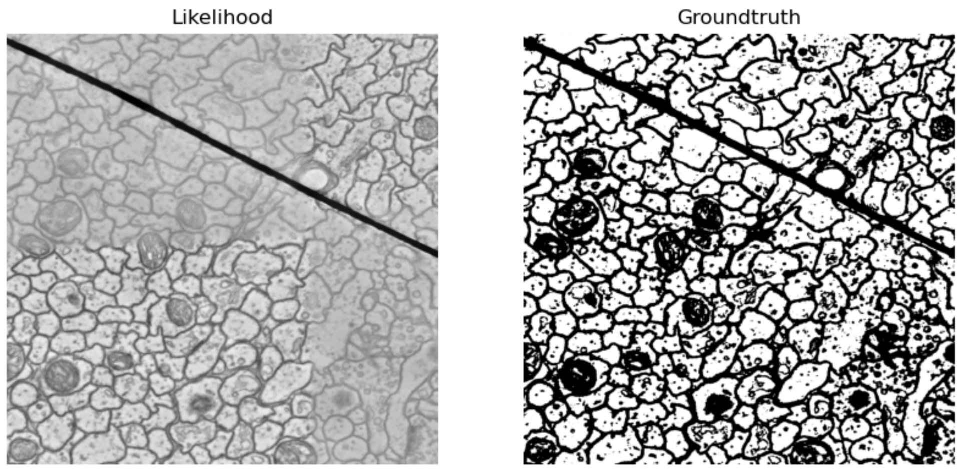


- Since it gets very crowded and to localize the features we take small patches of an image and work on it rather than working on one big image entirely.

Training Process and Modelling

Dataset

We use the CREMI dataset from here. This is a neuron image segmentation dataset. Here the volumes are stored in row-major format, i.e., with dimensions (depth,height,width). All volumes have an attribute resolution that specifies the size of voxels (depth,height,width) in nm. We use the volumes/raw which are gray scale pixels. There are 125 images of size (1250×1250). The images are in the form of arrays which go through normalization for better working. Below is example of one image whose likelihood and groundtruth are shown.



Example of an image used

Random Sampling

Every image before being passed into the network goes through Random Patch Sampling. Several (65×65) patches are sampled from an entire image randomly and every patch gets passed through the network during the training process. Train on small patches (65×65) to localize and correct topological structures. By focusing on smaller patches, we localize topological structures and fix them one by one.

Network Architecture

We use four different methods to make the neural network.

1. There are a total of 6 trainable layers: four convolutional layers and two fully connected layers. First 4 layers are convolutional layers out of which 1st, 2nd and 4th layer are followed by 2x2 max pooling with a stride of 2. 5th and 6th layers are fully connected layers. It is designed to extract hierarchical features from input images while preserving spatial relationships. It includes fully connected layers at the end for classification.

2. Based on the VGG19 architecture, this model has been pretrained on a large dataset (likely ImageNet) and then fine-tuned for our specific task. It retains the deep convolutional feature extraction layers of VGG19 but adapts the final layers (including custom convolutional and fully connected layers) to the specific requirements of the new dataset, enhancing performance through transfer learning.

3. This enhanced CNN model incorporates batch normalization after each convolutional layer, which stabilizes and accelerates training by normalizing the input to each layer. It also includes dropout regularization before the final fully connected layers to prevent overfitting. The additional convolutional and pooling layers further deepen feature extraction, making it well-suited for complex image recognition tasks requiring robust feature hierarchies and reduced overfitting.

4. In the fourth model, we do not incorporate the Topological Loss, and therefore, we do not utilize the Persistence Diagram. We employ the enhanced CNN model described previously and incorporate only the Cross-Entropy Loss.

Metrics used

- **Betti Number Error**
  The Betti number error measures the topological accuracy of segmentation by comparing the Betti numbers (a count of connected components, holes, and voids) between the predicted segmentation and the ground truth. It ensures that the segmented output preserves the topological features of the target structure.

$$\text{Betti Number Error} = \sum_{i=0}^{d} |\beta_i^{\text{pred}} - \beta_i^{\text{true}}|$$

  where β_i^{pred} and β_i^{true} are the i-th Betti numbers of the predicted segmentation and the ground truth, respectively.

- **Per-Pixel Accuracy**
  Per-pixel accuracy is a straightforward metric for evaluating segmentation performance by comparing each pixel's predicted label with the ground truth label. It is the ratio of correctly predicted pixels to the total number of pixels.

$$\text{Per-Pixel Accuracy} = \frac{\text{Number of Correct Pixels}}{\text{Total Number of Pixels}} = \frac{\sum_{i=1}^{N} \mathbf{1}(y_i = \hat{y}_i)}{N}$$

  where N is the total number of pixels, y_i is the ground truth label, $\hat{y}_i$ is the predicted label, and $\mathbf{1}(\cdot)$ is the indicator function.

- **Variation of Information**:
  Variation of Information measures the distance between two clusterings (segmentations) by quantifying the amount of information lost and gained in transitioning from one clustering to another. It combines measures of entropy and mutual information.

$$\text{VI}(X, Y) = H(X) + H(Y) - 2I(X; Y)$$

Results

1. Base Model with Topological Loss

Epoch	**Loss**	**Pixel Accuracy**	**Betti Error**	**VOI**
0	2.915325	0.7775	1.686191	0.505936
1	2.880974	0.7845	1.718177	0.496712
2	2.919731	0.7810	1.630852	0.501167
3	2.968226	0.7880	1.860091	0.494805
4	2.939657	0.7920	1.620122	0.487867
5	2.963847	0.7660	1.708620	0.528234
6	2.915276	0.7920	1.666932	0.491040
7	2.930497	0.7895	1.753002	0.488380
8	2.962167	0.7785	1.734487	0.504222
9	2.942901	0.7830	1.755237	0.503005

2. Fine-tuned VGG-19 model with Topological Loss

Epoch	**Loss**	**Pixel Accuracy**	**Betti Error**	**VOI**
0	2.903499	0.7705	1.676582	0.513870
1	2.934648	0.7810	1.712560	0.505603
2	2.957126	0.7760	1.721290	0.515304
3	2.905823	0.8065	1.678213	0.470138
4	2.909773	0.7905	1.795453	0.491604
5	2.903177	0.7990	1.775797	0.480397
6	2.939375	0.7820	1.679502	0.500900
7	2.964126	0.7755	1.688278	0.501794
8	2.932116	0.7810	1.703622	0.507570
9	2.998013	0.7785	1.667162	0.510092

3. Enhanced CNN model with Topological Loss

Epoch	**Loss**	**Pixel Accuracy**	**Betti Error**	**VOI**
0	2.932299	0.7910	1.700360	0.492736
1	2.893417	0.7910	1.646253	0.486072
2	2.960506	0.7810	1.740071	0.507909
3	2.948113	0.7810	1.611626	0.507099
4	2.925839	0.7805	1.680965	0.501540
5	2.961712	0.7575	1.695975	0.532899
6	2.902248	0.7875	1.693363	0.493086
7	2.866128	0.7830	1.834988	0.498587
8	2.851740	0.7935	1.758840	0.487646
9	2.953988	0.7740	1.628758	0.513936

4. Enhanced CNN model without Topological Loss

Epoch	**Loss**	**Pixel Accuracy**	**Betti Error**	**VOI**
0	0.681954	0.7660	0.0	0.527386
1	0.680024	0.7915	0.0	0.487441
2	0.680635	0.7735	0.0	0.521279
3	0.680762	0.7695	0.0	0.518605
4	0.681699	0.7725	0.0	0.515658
5	0.680213	0.7775	0.0	0.515984
6	0.677606	0.8000	0.0	0.468761
7	0.680464	0.7850	0.0	0.496151
8	0.679566	0.7830	0.0	0.504454
9	0.680274	0.7900	0.0	0.494208

For all methods, we generate segmentations by thresholding the predicted likelihood maps at 0.5. Our loss is a weighted combination of cross entropy loss and topological loss. For convenience, we drop the weight of cross entropy loss and weight the topological loss with λ.

Conclusion

The results from the four models reveal varying performance metrics in terms of loss, pixel accuracy, Betti error, and variation of information (VOI).

- The Base Model with Topological Loss demonstrates balanced metrics but does not lead in any particular category.
- The Fine-tuned VGG-19 model with Topological Loss shows an improvement in pixel accuracy, peaking at 0.8065, and a lower Betti error, indicating better structural preservation.
- The Enhanced CNN model with Topological Loss displays consistent pixel accuracy and improved Betti error, highlighting its ability to capture fine structural details.
- However, the model without Topological Loss, despite having a much lower loss value of around 0.68, suffers in terms of structural accuracy, as indicated by the zero Betti error. This is because the Betti Numbers depend on the Persistence of likelihood f and groundtruth g. Since we donot incorporate Topological Loss in the fourth model, we get BettiNumber Error as zero in all epochs.

Summary

Our topological loss compliments cross-entropy loss by combating sampling bias. Both of these losses are needed in order to get a succesful result.

In blurry or hard-to-see areas, making accurate predictions is much harder. These tough spots make up only a small part of the training samples. Adding more annotated images doesn't fix this imbalance. Topological loss helps by spotting these difficult areas during training, identifying critical pixels. It makes the network focus on learning patterns near these spots, which can cause some overfitting and a small drop in per-pixel accuracy. However, topological loss alone isn't enough to achieve the best results.

Without cross-entropy loss, trying to infer topology from a random likelihood map doesn't make much sense. Cross-entropy loss helps create a meaningful likelihood map, which allows the topological loss to effectively improve the map's topology.

Acknowledgement

I wish to extend my deepest appreciation to Dr. Amit Chattopadhayay, Professor at the International Institute of Information Technology Bengaluru, for his exceptional guidance and support throughout this research. His expert advice and constructive feedback were invaluable in the development of this work.

I am also profoundly grateful to the IASc-INSA-NASI Summer Research Fellowship Program for their generous support and for providing me with the opportunity to pursue this research over the course of two months. This fellowship was critical to the successful completion of the project.

My heartfelt thanks to all those who contributed to this research, both directly and indirectly, and made this endeavor possible.

References

[1] Herbert Edelsbrunner. *A Short Course in Computational Geometry and Topology.* Springer Publishing Company, Incorporated, 2014.

[2] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction.* American Mathematical Society, 01 2010.

[3] Xiaoling Hu, Fuxin Li, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[4] Roland Kwitt, Christoph Hofer, Andreas Uhl, and Marc Niethammer. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems 30 (NIPS)*, page 1633–1643, 2017.

[5] Agata Mosinska, Pablo Marquez-Neila, Mateusz Kozinski, and Pascal Fua. Beyond the pixel-wise loss for topology-aware delineation. *Computer Vision and Pattern Recognition*, 12 2017.

[6] Hubert Wagner, Chao Chen, and Erald Vuçini. Efficient computation of persistent homology for cubical data. *Homology and Topology*, 11 2012.