Modeling Hierarchical Topological Structure in Scientific Images with Graph Neural Networks

Samuel Leventhal
University of Utah *
samlev@cs.utah.edu

Attila Gyulassy
University of Utah
jediati@sci.utah.edu

Valerio Pascucci
University of Utah
pascucci@sci.utah.edu

Mark Heimann
Lawrence Livermore National Laboratory
heimann2@llnl.gov

Abstract

Topological analysis reveals meaningful structure in data from a variety of domains. Tasks such as image segmentation can be effectively performed on the network structure of an image's topological complex using graph neural networks (GNNs). We propose two methods for using GNNs to learn from the hierarchical information captured by complexes at multiple levels of topological persistence: one modifies the training procedure of an existing GNN, while one extends the message passing across all levels of the complex. Experiments on real-world data from three different domains shows the performance benefits to GNNs from using a hierarchical topological structure.

1 Introduction

Topological data analysis has been used in many application domains, such as segmentation of neurons (1), structural components of interest in metallic foams (2), eddies in ocean currents (3), bubble formation in mixing fluids (4), and ignition kernels in combustion (5). It has also served as a preprocessing step for machine learning tasks (6), and topological principles have influenced the design of loss functions (7) and architectures (8; 9) for deep neural networks.

In each case, semantic objects appear as elements (or collections thereof) of the data structures encoding the topological abstraction, for example as nodes within a graph encoding of a topological complex. We thus formulate the task of scientific image segmentation as a node classification problem, which we solve using graph neural networks (GNNs). Instead of deriving only one high-granularity complex, we model multiple scales of topological information in an image using a nested hierarchy of graph representations, each derived from different levels of topological persistence. Our work makes the following contributions:

- We propose two different methods for applying GNNs to a hierarchy of topological graphs: one trains a conventional GNN on each successive level, and the other extends the message passing to take place jointly across all levels.
- We use GNNs to perform scientific image segmentation operating on the topological complex, for which we have developed an interactive tool to allow a human to label training data. We show a performance improvement by learning from hierarchical topological structure.

*Work partially performed while an intern at Lawrence Livermore National Laboratory

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2 Background: Computational Topology

The Morse-Smale Complex (MSC) A Morse function $f : \mathscr{M} \to \mathbb{R}$ is a smooth function on a manifold with non-degenerate, distinct critical points. The gradient, ∇f defines a vector field whose zeroes are critical points. Each non-critical point in the domain of $\mathscr{M}$ belongs to a single integral line, or path tangent to ∇f that has upper and lower limits at critical points of f (called the *destination*, and *origin*, respectively). Partitioning the domain into monotonic connected components defined by integral lines sharing a common origin and destination defines the MSC. *Cells* of this complex have a dimension equal to the difference between the number of source directions between the destination and origin critical points of their constituent integral lines. The *1-skeleton* of the complex is formed by critical points and the *arcs* or integral lines that connect them which differ in index by 1. Concepts from continuous functions can be applied to a discrete pixel space using discrete Morse theory (10), for which we use the open-source MSCEER library (11) to compute a discrete MSC complex. We provide an illustration of the (continuous and discrete) MSC in Appendix A.

Topological Simplification Topological abstractions come equipped with well-understood techniques to order and simplify their elements to obtain successively coarser representations. For example, *topological persistence* allows for a multiscale simplification by pairing the critical point that creates a connected component with the critical point that destroys that component during a filtration (12). The time span in the filtration in which the connected component lives, *i.e.* the difference in function value between the birth and death critical points, is called persistence. The admissible persistence defining this filtration defines the granularity of the resulting MSC. To simplify the MSC and minimize the effects of image noise, nodes connected by a single arc in the 1-skeleton are canceled (12; 13; 14).

Topological Priors Graph We represent an image with a refinement of its MSC 1-skeleton called the *ridge graph* (1), whose edges are poly-line segments of the MSC. Our *priors graph* is the line graph representation (15) of the ridge graph, in that vertices of the ridge graph become edges and polylines become nodes which are classified as foreground or background. This allows us to segment the image using its topological structure. We provide an illustration of the construction of the ridge graph and the priors graph we introduce in Appendix A.

Multi-Persistence Topological Graph Hierarchy To model multiscale topological information, we compute the MSC at P levels of persistence. In contrast to hierarchical graph-level representation learning methods that learn to pool each input graph (16; 17), this gives us several graph representations of the underlying data to use as input to a GNN or to inform the learning process for a GNN topologically. . Computationally, a smaller persistence value produces a graph higher in the MSC hierarchy with finer granularity. Thus, we obtain a hierarchy of graphs $G_1, \ldots, G_P$ where $\forall p_i$ for $i \in [1, \ldots, P], G_{p_{i-1}} \subseteq G_{p_i}$: that is, $G_{p_{i-1}}$ is an induced subgraph defined on a subset of the vertices in G_{p_i}. Node neighborhoods thus share this property: for each node $v_i \in \mathbf{V}_{p_i} \cap \mathbf{V}_{p_j}$, e.g. $\mathscr{N}_{p_i}(v_i)$ for $v_i \in \mathbf{V}_{p_i}$ and $\mathscr{N}_{p_j}(v_i)$ for $v_i \in \mathbf{V}_{p_j}$, we have that $\mathscr{N}_{p_i}(v_i) \subseteq \mathscr{N}_{p_j}(v_i)$. We give all graphs the full node set found at the lowest persistence level of the hierarchy, but nodes that would not otherwise exist in the graph at a higher persistence level are disconnected.

3 Methodology

3.1 Identifying and Labeling Topological Graph Hierarchies

A single topological primitive in a higher-level persistence subgraph may span multiple topological primitives in a lower-level persistence supergraph. Given a labeled priors supergraph, a topological primitive in a priors subgraph is assigned the majority class label assigned to primitives within the corresponding part of the supergraph as long the proportion of primitives in the supergraph exceeds a user-specified percentage which we call a union threshold. By doing so, finer granularity priors that more accurately cover the semantic object as represented in the supergraph are identified and labeled correspondingly in the subgraph. Formally, for persistence levels p_i and $p_j \geq p_i$, the class ℓ_j of a topological prior v_{p_j} in the priors subgraph of persistence p_j is subset to priors supergraph of persistence p_i with labeling scheme ℓ_i. Given a union threshold u the label ℓ_j is given as follows: $\ell_j = \ell_i$ if $\frac{1}{|v_{p_i} : v_{p_i} \in v_{p_j}|} \sum_{v_{p_i} \subseteq v_{p_j}} 1(\ell(v_{p_i}) = \ell_i) \geq u$. In short, each prior in the subgraph is given the

predominant label of priors in the supergraph that correspond to it, so long as the majority label of priors exceeds a user-specified threshold u.

3.2 Hierarchical Learning with Topological Priors Graphs

We consider two methods for learning node representations for classification using graph neural networks while exploiting the full hierarchy of topological information. The first method modifies the training of a graph neural network (GNN), while the second modifies the architecture to pass messages jointly on all hierarchical levels. With both approaches, each GNN learns node features via message passing in the spatial domain (18). For the ith persistence level p_i (moving from sparsest to densest), the aggregated node embedding for node v_i from neighbors $u_i \in \mathcal{N}_{p_i}(v_i)$ of persistence subgraph G_i is given by $h^{k_i}_{\mathcal{N}_{p_i}(v_i)} = \sigma\big(AGGR(h^{k_i-1}_{u_i} \ : \ \forall\, u_i \ \in \ \mathcal{N}_{p_i}(v_i)\ )\ \big)$. Node v_i's updated embedding is concatenated with the target node embedding from the previous iteration of aggregation $h^{k_i}_{v_i}$ and the aggregated neighbor features, parameterized with target embedding and neighbor embedding weight matrices $W_s^{k_i}$ and $W_n^{k_i}$ respectively. Similarly, the target node embeddings and neighboring node embeddings are passed through a two-layer multilayer perceptron (MLP), which after sum-pooling, are multiplied with weight matrices W_s and W_n to complete the AGGR step.

Hierarchical Successive Training (HST) For HST we exploit the induced subgraph structure of the persistence graph hierarchies and take the naive approach of aggregating node embeddings between subgraphs by iteratively introducing higher-level subgraphs in later training epochs. Instead of training a GNN for N epochs on the finest graph G_P, we train it for $\frac{N}{P}$ epochs each on graphs $G_1, \ldots, G_P$, using the embeddings from G_{i-1} to initialize the embeddings of corresponding nodes in $G_i \ \forall\, i \in [1 \ldots, P]$. The MLPs and node embedding weight matrices for target and neighboring node embeddings are shared between graph hierarchies. Training begins with the smallest subset graph and iteratively increases in graph size. We also successively share the learned embedding weight matrices and matrices of the MLPs as well. When we begin training on graph G_p, the embedding for node $v_p \in \mathcal{V}(G_p)$ is the combined embedding from previous aggregations at lower persistence subgraphs: $h^{k_p}_{v_p} = COMB\big(h^{k_{p-1}}_{v_{p-1}}, h^{k_p-1}_{v_p}\big)$ where we implement COMB with concatenation.

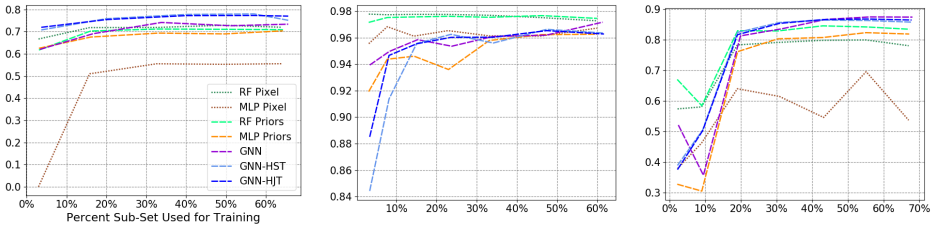
Hierarchical Joint Training For node v_i belonging to node set $\mathbf{V}_{p_i}$ of subgraph $\mathbf{G}_{p_i}$, the feature representation $h^k_{v_i}$ at GNN layer $k \in \{1, \cdots, K\}$ first combines self-representations from the previous level of aggregation with aggregated neighbor information: $h^k_{v_i} = COMB\big(\ W_s^{k-1}\ h_v^{k-1}\ ,\ AGGR\big(\{W_n^{k-1} h_u^{k-1}\ :\ u\ \in \mathcal{N}_{p_i}(v_i)\}\big)\big)$. Once this is done, the resulting embedding is combined with the embedding representation from other persistence subgraphs, that is, node embeddings culminate from message passing between levels in the graph hierarchies. As an example for two persistence levels p_i and p_j where $v_i \in \mathbf{V}_{p_i} \subset \mathbf{G}_{p_i}$ and $v_j \in \mathbf{V}_{p_j} \subset \mathbf{G}_{p_j}$, we have $h^k_v = \sigma\big(COMB(h^k_{v_i}, h^k_{v_j})\big)$.

4 Experiments

Data We use image datasets from three different application domains: biomedicine, neuroscience, and materials science. For each image, to optimize the ridge/valley graph and corresponding topological priors so as to best cover the semantic object, a functional operator is first applied to the image. Performing this pre-processing step allows for a topologically informative scalar field representation of the image, or scalar field image, allowing for a more robust and expressive MSC summary of the target semantic object. For more details about each dataset, see Appendix B. The foreground is labeled by a human expert using an interactive tool that we have developed, described in Appendix C, where a user can also provide a scalar field image deemed informative. As a basic demonstration of a topological hierarchy, we use two levels of persistence in total (resulting in one subgraph and one supergraph). We choose persistence levels per dataset such that the subgraph was sufficiently smaller in size and sufficiently covered the semantic object (see Table 1 in Appendix B for specific persistence values.)

Metrics We study several models for learning from pixel-level and topological features, detailed in Appendix D, and compare their class F_1 scores. Training regions for each image are grown and first centered in areas that illustrate the diversity of canonical representatives of the semantic objects and train over potential confounding artifacts or morphologies. For details on the training procedure and parameter selection, see Appendix E.

Evaluation Figure 1 shows that our hierarchical approaches (**GNN-HST** and **GNN-HJT** for hierarchical successive training and joint training, respectively) generally improve over the conventional **GNN** and non-GNN baselines. This is particularly true for the Neuron and Retinal datasets, where there is a larger difference between the topological summary of the computed subgraphs and supergraphs (this can be seen by comparing their number of nodes and edges, shown in Appendix B).

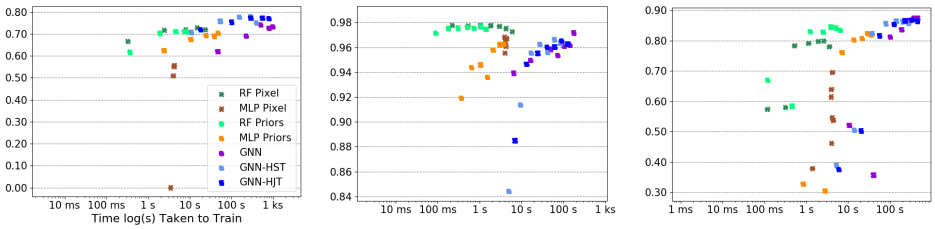


(a) Neuron (b) Foam (c) Retinal

Figure 1: Class F_1 vs. percent priors graph used in training. We see improvements from hierarchical training on the Neuron and Retinal datasets with lower performance on Foam (perhaps due to larger class imbalance and/or less informative topological priors).

Limitations: Hierarchical GNN training was less effective on the Foam dataset. On this dataset, methods performing learning in the pixel space (Appendix D) achieved higher performance than learning with topological priors. This may be due to the sparsity of the priors graphs and class imbalance (see Appendix F for further discussion) or due to the simplicity of the segmentation task for the Foam dataset leading low and high-level persistence priors graphs to vary marginally in comparison to that of the Neuron and Retinal datasets. Future work should seek to understand how well topological priors model semantic structure and how to represent informative topological information.

Our GNN-HST variation also enjoys runtime gains compared to the conventional GNN using the full supergraph for all of the training, due to the successive training scheme allowing us to perform a portion of training on the smaller subgraph. Our joint training yields slightly higher accuracy as the model simultaneously uses multiple scales of topological information, and the runtime remains manageable. While non-network baselines, using famously fast ML models such as random forest and few-layer MLPs (Appendix D), are faster, we have seen that they generally offer lower accuracy.



(a) Neuron (b) Foam (c) Retinal

Figure 2: Class F_1 vs. time taken to train priors graph used in training. Times and accuracies correspond to training region size and accuracy as provided in Figure 1. Among the GNN approaches, the hierarchical methods, notably the successive training, perform the fastest.

5 Conclusion

We propose graph neural network methods that learn from hierachies of graphs representing the multiscale topological structure of image data. Our results demonstrate promise for increased accuracy and improvement in training time compared to conventional GNNs, while we provide exploratory insights into the effect of class imbalance and heterophily in graph learning.

Social Impact Our methods have the potential to minimize the amount of human effort required to extract semantic structure from scientific images, which may contribute to socially beneficial breakthroughs in medicine, materials science, and more. As training data comes from human annotations, our methods are subject to reproducing annotators' biases. A socially impactful future direction will be to better understand how topological priors reflect semantic structure as well as extend the domain with which we can construct priors graphs by using higher-dimensional topological connected components.

References

[1] T. McDonald, W. Usher, N. Morrical, A. Gyulassy, S. Petruzza, F. Federer, A. Angelucci, and V. Pascucci, "Improving the usability of virtual reality neuron tracing with topological elements," *IEEE transactions on visualization and computer graphics*, vol. 27, no. 2, pp. 744–754, 2020.

[2] S. Petruzza, A. Gyulassy, S. Leventhal, J. J. Baglino, M. Czabaj, A. D. Spear, and V. Pascucci, "High-throughput feature extraction for measuring attributes of deforming open-cell foams," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 140–150, 2019.

[3] H. Xue and Y. Gu, "Application of topological analysis in ocean feature extraction," *Computer Engineering*, vol. 35, no. 3, p. 263, 2009.

[4] V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*. Springer Science & Business Media, 2010.

[5] P.-T. Bremer, W. Cabot, A. Cook, D. Laney, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities," in *Journal of Physics: Conference Series*, vol. 46, p. 077, IOP Publishing, 2006.

[6] S. Banerjee, L. Magee, D. Wang, X. Li, B.-X. Huo, J. Jayakumar, K. Matho, M.-K. Lin, K. Ram, M. Sivaprakasam, *et al.*, "Semantic segmentation of microscopic neuroanatomical data by combining topological priors with encoder–decoder deep networks," *Nature Machine Intelligence*, vol. 2, no. 10, pp. 585–594, 2020.

[7] X. Hu, Y. Wang, L. Fuxin, D. Samaras, and C. Chen, "Topology-aware segmentation using discrete Morse theory," *arXiv preprint arXiv:2103.09992*, 2021.

[8] M. Moor, M. Horn, B. Rieck, and K. Borgwardt, "Topological autoencoders," in *International conference on machine learning*, pp. 7045–7054, PMLR, 2020.

[9] Q. Zhao, Z. Ye, C. Chen, and Y. Wang, "Persistence enhanced graph neural network," in *International Conference on Artificial Intelligence and Statistics*, pp. 2896–2906, PMLR, 2020.

[10] R. Forman, "A user's guide to discrete Morse theory," *Sém. Lothar. Combin*, vol. 48, p. 35pp, 2002.

[11] A. Gyulassy, "MSCEER: Morse-Smale complex extraction, exploration, reasoning," 2018.

[12] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," in *Proceedings 41st annual symposium on foundations of computer science*, pp. 454–463, IEEE, 2000.

[13] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci, "A multi-resolution data structure for two-dimensional Morse-Smale functions," in *IEEE Visualization, 2003. VIS 2003.*, pp. 139–146, 2003.

[14] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann, "A topological approach to simplification of three-dimensional scalar functions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 474–484, 2006.

[15] Z. Chen, L. Li, and J. Bruna, "Supervised community detection with line graph neural networks," in *International Conference on Learning Representations*, 2019.

[16] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in neural information processing systems*, vol. 31, 2018.

[17] H. Gao and S. Ji, "Graph u-nets," in *international conference on machine learning*, pp. 2083–2092, PMLR, 2019.

[18] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

[19] E. Van der Merwe and S. Kidson, "Advances in imaging the blood and aqueous vessels of the ocular limbus," *Experimental eye research*, vol. 91, no. 2, pp. 118–126, 2010.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[21] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the 35th International Conference on Machine Learning, ICML*, vol. 80, pp. 5449–5458, PMLR, 2018.

[22] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," in *NeurIPS*, 2020.

A Illustration of the Morse-Smale Complex

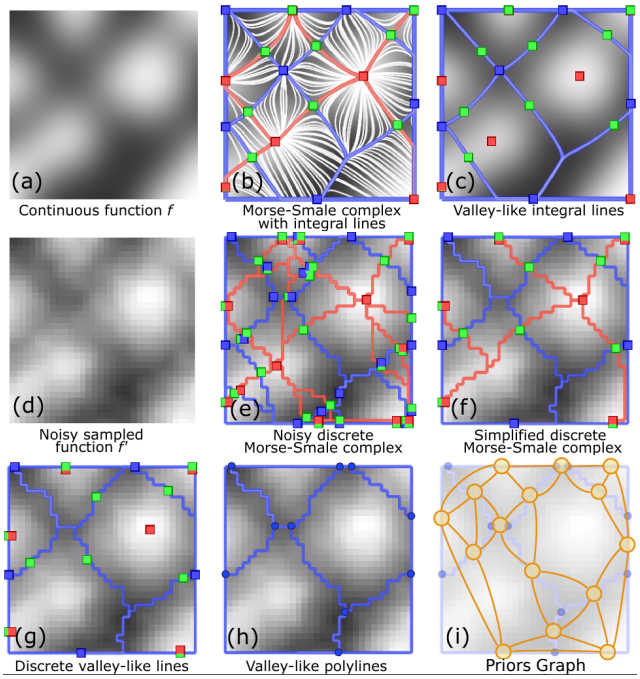


Figure 3: Morse-Smale complexes are defined for functions with continuous gradients **(a-c)**. A smooth function **(a)** can be partitioned based on the behavior of integral lines **(b)**, with selected integral lines shown in white. This partition forms a cell complex, where integral lines within each cell share a common origin and destination. The 0-dimensional cells are maxima (red), saddles (green), and minima (dark blue)), the 1-dimensional cells are formed by ascending (orange) and descending lines (light blue) from saddles (green), and 2-dimensional cells are bounded by 0- and 1-cells **(b)**. Elements of this complex often form semantic features of interest in a scientific domain, such as valley-like lines **(c)**. Real-world functions often come from noisy sources, and are available as samples on a grid **(d)**. Discrete Morse-theory-based methods allow practical computation of Morse-Smale complexes **(e)**, which encode both noise and discretization artifacts that may be simplified to recover the coarse-scale behavior of the function **(f)**. The valley-like structures may be extracted from this complex **(g)**, and converted to a set of priors between non-degree-2 vertices denoted the valley graph **(h)**. The priors graph (yellow), **(i)**, represents each prior as a vertex with edges between incident priors.

B Detailed Description of Datasets

Here we give more detailed descriptions of the three datasets we use for evaluation, along with some summary statistics in Table 1.

Table 1: Dataset and Priors Graph Statistics

Name	Image Shape	|V| Sub/Supergraph	|E| Sub/Sup	p_i Sub/Sup	Total Length	% Fgrnd.
Retinal	700×605	676 / 24,851	1,097 / 39,765	0.8 / 0.01	269,036	11.6%
Foam	828×846	1,703 / 8,069	2,429 / 13,234	80 / 20	142,895	69.9%
Neuron	$1,737 \times 1,785$	323 / 23,038	480 / 34,527	45 / 11	425,441	15.5%

Retinal: Imaging and tracing of blood vessel arbors is used to classify disease states of the eye (19). Obtaining a wire representation of the blood vessels is a first diagnostic step. For this we use the Laplacian of the image with a kernel of size 2.0, producing a scalar field whose ridge lines, and priors graph, capture even the faintest of blood vessels.

Foam: A computed tomography (CT) image of a closed-cell foam is used to characterize the deformation of cell walls that may result from its manufacturing process. The thin film polymer boundaries are faint compared to the background CT noise. We compute the maximum convolution with a rotating line 10px in length and a total of 16 directions to enhance linear structures and smooth noise. The topological priors of this field include both cell boundaries and noise.

Neuron: Viral expression fluorescent proteins, combined with tissue clarification techniques, allow imaging of sparse neurons and their projections. Understanding neural circuitry, neurons and their constituent dendritic and axonal subtrees, is a central task in a wide range of biomedical and neuroscience applications. We use a Gaussian smoothed image with a kernel size of 2.0 to generate topological priors.

C Fast User Annotation of Topological Priors in 2D

We have developed an interactive tool to facilitate and accelerate labeling of priors graphs. A screenshot of the painting is shown in Figure 4. The tool uses spatial acceleration structures to clamp the user's interaction to the nearest relevant topological element, and uses shortest path algorithms to facilitate drawing long paths, branching trees, and closed loops. Flood fill and region selection tools allow rapid labeling of homogeneous regions.

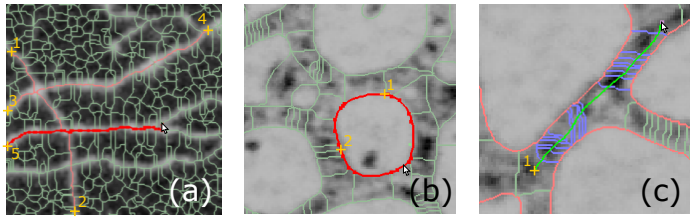


Figure 4: Interactive tool allowing a human to specify a segmentation by labeling topological priors, specifically polylines, as opposed to individual pixels. With the shortest-path tool, a user is six clicks into labeling the foreground neurons (a). A user only needs 3 clicks to draw a closed loop (b). Finding objects crossing a free-form stroke allows rapid labeling (c).

D Feature Statistics and Models

An image $\mathbf{X}$ is a 2-D tensor whose i,j-th entry $\mathbf{X}_{ij}$ represents the value (e.g. grayscale intensity) of pixel (i,j) in that image. Images are first normalized prior to feature extraction in which additional features for each pixel (i,j) are computed. We apply a series of transformations to the image, each modeled as a function $f(\mathbf{X})$ that returns an image of the same size, and consider the values $f(\mathbf{X})_{ij}$ for various functions f as additional features for pixel (i,j). We create features of dimension $d = 47$ by using d different choices of image transformation functions f. Our goal here is not to perform exhaustive feature engineering, but to provide samples from widely used techniques. Our transformations include the identity of the image, Gaussian blurring and differences of Gaussians with standard deviation $\sigma \in \{2,4,8,16\}$ to serve as a smoothing kernel for capturing pixel neighborhood information, the maximum, minimum, median, and variance of pixel neighborhoods for radius sizes $\{2,4,8,16\}$, Sobel filtration, Gaussian edge detection, and Hessian eigenvalue filtration.

Nodes having originated from topological priors also afford aggregate pixel statistics for those covered by priors. For these we compute the median, minimum, maximum, standard deviation, and variance among pixel intensities. If the original pixels then have $d-$dimensional features, the priors are represented by a $5 \times d-$dimensional feature vector resulting in 235 features per topological prior.

We train a random forest and multilayer perceptron (MLP) on these pixel-level feature statistics, which we denote by **RF Pixel** and **MLP Pixel**. We also investigate the performance of pixel-level classification using a U-Net (20), which takes considerably longer to train.

We construct features for each topological prior using the aggregate statistics (median, minimum, maximum, standard deviation, and variance) for pixel intensities of the pixels covered by the geometry of the topological prior. We again perform classification in this space of topological priors (without

Table 2: Homophily and class balance ratio for subgraph and entire graphs.

Data	Homophily Ratio		Class Ratio	
	Subgraph	Full Graph	Subgraph	Full Graph
Neuron	0.35	0.86	1.32	0.28
Foam	0.38	0.76	5.12	1.93
Retinal	0.39	0.91	0.55	0.16

using the graph structure) using a random forest and a MLP, denoted **RF Priors** and **MLP Priors**. To use the graph structure as well as the topological priors' features, we use GraphSAGE (18) as a baseline **GNN** model to learn node representations. We denote our hierarchical GNN methods as **GNN-HJT** and **GNN-HST** for hierarchical joint training and hierarchical successive training, respectively.

E Experimental Procedure

Hyperparameters We performed a parameter sweep of learning rates $\{1,2,3\} \times \{1\mathrm{e}^{-2}, 1\mathrm{e}^{-3}, 1\mathrm{e}^{-4}\}$ finding $3e^{-3}$ to be the best. We trained each model for 10 epochs following the original GraphSAGE paper (18) and use a weight decay of $1e^{-7}$. For each variant of graph neural network we use hidden vertex embedding dimensions of 512 and 1024 with output vertex embedding of 256, and aggregate neighbors' embeddings by maximum pooling. We also add jumping knowledge between aggregation layers of the GNN (21), which has been shown beneficial to combat high class heterophily that our datasets exhibit (22), leading us to use four layers total.

Computing Environment All experiments were run on a laptop with 3 GB GeForce GTX 970M with 1280 CUDA Cores GPU and 3.5 GHz i7-6700HQ (2.6 up to 3.5 GHz – 6MB Cache – 4 Cores – 8 Threads) processor running Ubuntu, Linux.

Training and Inference Procedure For each model and inference run, we perform a parameter sweep for the foreground/background probability threshold to maximize the class F_1 score - which directly correlates to the Dice score and is functionally equivalent to the mean IOU class score.

For training, we chose sub-regions that accurately capture the diversity of geometric information and variability within each dataset, starting with size 64×64. We then grow the training boxes by approximately 10% of the image until terminating once the percentage of training exceeded more than 60% of the image. Topological priors belonging to the priors graph entirely within the regions of intersecting tiles of size 64×64 was used for training, and the remainder for inference probabilities are assigned to all priors as belonging to the foreground target object class.

F Homophily and Class Balance

To gain an understanding of how homophily and class imbalance may affect the performance of the methods we study, in Table 2 we report the homophily and class balance ratios for the priors graphs (at higher and lower levels of persistence) derived from the datasets we consider. The class balance ratio is defined as the number of total nodes positively labeled as foreground over the total number of negatively labeled. For nodes u and v with labels y_u and y_v and edge set $\mathscr{E}_i$ for graph G_i in the persistence hierarchy, the homophily ratio (22) is given by $h_i = \frac{1}{|\mathscr{E}_i|} \sum_{v \in \mathscr{E}_i} \frac{|\{u \in \mathscr{N}(v) : y_u = y_v\}|}{|\mathscr{N}(v)|}$

Intuitively, we expect the class imbalance to be larger in the subgraphs due to being associated to sparser persistence subgraph. Namely, the ridge/valley graph from which these graphs originate spans larger "ridge like" connectivity regions in the image space which are more likely, and with less accuracy, associated to the semantic object. We anticipate the homophily ratio to be lower for this same reason for lower-level persistence graphs since the loss of precision in expansive topological priors may result in a mixture of adjacent class labels in the priors graph. Note that for the Foam dataset, class imbalance is much higher. This may be one reason for our hierarchical methods' low relative performance on this dataset (and indeed the lower performance of all graph-based methods), to be explored in future work.