

---

# HyperTopo-Adapters: Geometry- and Topology-Aware Segmentation on Frozen Encoders

---

**Alice A. Author**

Department of Computer Science, University of Wonderland  
Wonderland, USA  
alice.author@wonderland.edu

**Bob B. Author**

Trident AI Research, Trident Corporation  
Atlantis, UK  
bob.author@trident.com

**Charlie C. Author**

Institute of AI Research, Example University  
Metropolis, Germany  
charlie.author@example.edu

## Abstract

Modern segmentation methods often rely on large pretrained encoders, yet fine-tuning these encoders for downstream tasks can be inefficient and may ignore domain-specific geometric and topological priors. We propose *HyperTopo-Adapters*, a framework for accurate image segmentation that attaches lightweight geometry- and topology-aware adapter modules to a frozen pretrained encoder. Our approach incorporates a manifold adapter head operating jointly in hyperbolic, Euclidean, and spherical spaces to capture multiscale geometry, a hyperbolic contrastive loss to reinforce hierarchical feature consistency, and a differentiable topology surrogate loss based on persistent homology to enforce topologically correct predictions. Experiments on challenging segmentation benchmarks demonstrate that HyperTopo-Adapters achieve competitive pixel-wise accuracy while significantly improving topological correctness (e.g., ensuring the right number of connected components and holes) compared to traditional fine-tuning. By leveraging both geometric embeddings and topological priors, our method produces segmentations that better preserve the structure of target objects. We hope our work sparks further research into integrating modern topology-aware constraints with foundation model adapters for computer vision.

## 1 Introduction

Deep neural networks have achieved remarkable success in image segmentation, from medical imaging to autonomous driving. Classical fully-supervised architectures such as U-Net ((13)) and DeepLabv3 ((1)) learn rich representations of visual features, but training them from scratch requires large labeled datasets. Meanwhile, pretrained backbone models (e.g. ResNet-50 ((5)) or recent vision transformers) have emerged as powerful feature encoders. Self-supervised visual transformers like DINOv2 ((11)) and DINOv3 ((14)) produce versatile, high-level feature maps that can be repurposed for downstream tasks without full finetuning. Adapting such frozen encoders via lightweight modules (as in adapter networks or LoRA ((6))) is an attractive strategy: it greatly reduces trainable parameters and avoids catastrophic forgetting in the pretrained features.

However, simply adapting a pretrained encoder for segmentation may yield outputs that are topologically inconsistent or geometrically suboptimal. For instance, standard loss functions (e.g. cross-entropy) optimize pixel-wise accuracy but do not guarantee that predicted segmentation masks have the correct number of connected components or holes. Topological errors—such as fragmented

structures or spurious holes in organs—can severely undermine the utility of segmentation in downstream analysis. Prior works have proposed to incorporate topological constraints via differentiable losses based on persistent homology ((2; 7)), showing improved structural accuracy. At the same time, there is growing interest in non-Euclidean embeddings (hyperbolic ((9)) or spherical) which can better capture hierarchy or boundary geometry in data. We hypothesize that marrying these geometry-aware representations with topological reasoning will lead to segmentations that are both accurate and structurally faithful.

In this paper, we introduce **HyperTopo-Adapters**, a novel segmentation approach that integrates geometric and topological awareness into a parameter-efficient adapter for frozen encoders. Our contributions are three-fold. **(1)** We design a *manifold adapter head* operating in a product space of hyperbolic, Euclidean, and spherical ( $H \oplus E \oplus S$ ) embeddings. This head augments pretrained features with richer geometric representations, enabling the model to capture global shape (hyperbolic for hierarchies) and boundary circularity (spherical) alongside local detail (Euclidean). **(2)** We introduce a *hyperbolic InfoNCE loss* that encourages consistent hierarchical features: using contrastive learning in hyperbolic space, we pull together embeddings of the same object or region under perturbations and push apart different objects. This regularizer reinforces the encoder’s representation without updating its weights. **(3)** We formulate a *topology surrogate loss* based on persistent homology computations, which penalizes discrepancies in the persistence diagrams of predicted and ground-truth masks. This loss provides direct feedback on topological errors (e.g. extra components or missing holes) in a differentiable manner. Together, these innovations allow HyperTopo-Adapters to produce segmentations that maintain both high overlap accuracy *and* correct topological structure.

We validate our approach on several segmentation benchmarks, demonstrating that our method yields superior topological correctness over baselines. HyperTopo-Adapters achieve comparable or better Dice and IoU scores than full-finetuning baselines while significantly reducing topological violations (as measured by Betti number errors and boundary F1). Qualitatively, our results show contiguous, intact segmentations that respect known anatomical topology (for medical images) or object shape continuity (for natural images). To our knowledge, this is the first work to integrate hyperbolic representations and persistent homology in a segmentation adapter for frozen models. We believe this opens a new direction for building structure-aware vision models by combining the strengths of geometric deep learning and topological data analysis.

## 2 Related Work

**Segmentation Architectures.** Fully convolutional networks and encoder-decoder models have long dominated image segmentation. U-Net ((13)) pioneered the encoder-decoder with skip connections for biomedical segmentation, achieving accurate pixel classifications with limited data. Later architectures like DeepLabv3 ((1)) improved semantic segmentation using atrous convolutions and multi-scale context. These models typically train from scratch or with ImageNet-pretrained backbones (e.g. ResNet-50 ((5))). In recent years, vision transformers and pyramid fusion networks have further advanced the state-of-the-art. Our work differs by focusing on adapting a *frozen* powerful encoder rather than designing a new encoder. We demonstrate that with appropriate adapters and losses, even a fixed high-level feature extractor can yield excellent segmentation results.

**Pretrained Encoders and Adapters.** Leveraging large pretrained models for downstream tasks has become standard in NLP and increasingly in vision. Self-supervised models like DINO ((11; 14)) produce generic feature representations that can be reused for detection or segmentation. Prior works have shown the benefit of using frozen transformers with lightweight task-specific heads (e.g. the Segmenter and Mask2Former approaches use pretrained ViT encoders). Parameter-efficient finetuning techniques, such as adapter layers and LoRA ((6)), insert small trainable components (bottleneck MLPs or low-rank matrices) into a model to adapt it without updating all weights. While most adapter methods have been explored in NLP or classification, we extend the idea to segmentation. HyperTopo-Adapters attach only a small number of trainable parameters to a frozen encoder, but crucially incorporate geometric (hyperbolic/spherical) transformations and topological constraints, which traditional adapters lack.

**Hyperbolic and Non-Euclidean Representations.** Hyperbolic embedding spaces have gained attention for representing hierarchical or graph-structured data, since distances in hyperbolic space

grow exponentially, naturally fitting tree-like structures ((9)). Hyperbolic neural networks and layers have been used for classification and knowledge graphs, and a few works have applied hyperbolic embeddings to computer vision tasks, e.g. image classification with class hierarchies. Spherical embeddings (points on a sphere) can be useful for directional data or enforcing normalized feature vectors. Our  $H \oplus E \oplus S$  manifold head draws inspiration from recent advances in geometric deep learning: by combining multiple geometries, the model can capture different aspects of structure. Specifically, the hyperbolic component is expected to encode the inclusion hierarchy of regions (e.g. capturing one component containing smaller parts), the spherical component can model angular relationships (like curvature of boundaries), and the Euclidean component retains standard linear feature interactions.

**Topology-Aware Segmentation.** There is a rich literature on imposing topological priors in segmentation. Early approaches ensured topological correctness via post-processing (morphological operations or contour smoothing). Recent methods explicitly incorporate topological terms into the loss function. For example, Clough *et al.* ((2)) introduced a loss that encourages predicted masks to have specified Betti numbers (counts of components and holes) by using persistent homology to compare with a target topology. Hu *et al.* ((7)) proposed *TopoLoss*, which computes the persistence diagram of the predicted probability map and ground truth, then penalizes differences via a matching cost. These approaches leverage tools from computational topology ((4)), often using libraries like GUDHI ((15)) to compute persistence diagrams. The main challenge is that topological computations are not naturally differentiable; both Clough and Hu devise differentiable surrogates or approximations. Our work builds on these ideas by incorporating a persistent-homology-based loss into the training of a segmentation adapter. Unlike previous works which trained entire CNNs with a topo loss, we show that even with a strong frozen encoder, adding a topology loss significantly improves segmentation structure. Furthermore, our method uniquely combines the topological loss with geometric feature learning (hyperbolic contrastive learning), which to our knowledge has not been explored in prior segmentation methods.

### 3 Method

Our goal is to perform segmentation by attaching a small trainable module to a large pretrained encoder, while injecting geometric and topological awareness to guide the training. An overview of the HyperTopo-Adapters architecture is shown in Figure 1. We use a frozen backbone  $\mathcal{F}$  (e.g. a ViT trained with DINOv2) to extract a spatial feature map from the input image. The adapter head  $\mathcal{A}$  then maps these features into a combined manifold space (Section 3.1). A lightweight decoder  $\mathcal{D}$  upsamples and transforms the adapter output into a segmentation mask (Section 3.2). The model is trained with a multi-component loss: a standard segmentation loss (cross-entropy or Dice loss), plus a hyperbolic InfoNCE contrastive loss (Section 3.3) and a topology surrogate loss (Section 3.4). We next describe each component in detail.

#### 3.1 $H \oplus E \oplus S$ Manifold Adapter Head

The adapter head  $\mathcal{A}$  takes as input the feature map from the last layer of the frozen encoder. Assume the encoder outputs feature tensors of shape  $H_e \times W_e \times C$  (for a CNN) or a set of patch embeddings of length  $N$  (for a ViT). We first project these features to a lower-dimensional space suitable for our manifold transformations. Specifically,  $\mathcal{A}$  consists of three parallel branches:

- **Hyperbolic branch:** A linear layer (or MLP) outputs  $d_H$ -dimensional vectors. These are interpreted as coordinates in the Poincaré ball model of  $d_H$ -dimensional hyperbolic space. We apply a nonlinearity (tanh) and rescale to ensure the vector norm is  $< 1$ , placing it inside the unit ball. The result  $z^H$  lies in  $\mathbb{B}^{d_H}$  with hyperbolic metric.
- **Euclidean branch:** A linear layer outputs a  $d_E$ -dimensional vector  $z^E$  (with no special constraints), representing standard Euclidean features. This branch essentially passes through information in a conventional vector space.
- **Spherical branch:** A linear layer outputs a  $d_S$ -dimensional vector, then we  $\ell_2$ -normalize it to lie on the  $d_S$ -dimensional unit sphere  $\mathbb{S}^{d_S-1}$ . The resulting  $z^S$  lives on a spherical manifold, capturing angular relationships.

figures/architecture\_schematic.pdf

Figure 1: **HyperTopo-Adapters Architecture.** A frozen encoder (e.g. DINOv2 ViT ((11))) produces a feature map. Our manifold adapter head projects features into hyperbolic, Euclidean, and spherical subspaces and combines them ( $H \oplus E \oplus S$ ). A decoder uses these enriched features to predict the segmentation. Training involves three losses: standard segmentation loss (e.g. cross-entropy), hyperbolic InfoNCE contrastive loss to align representations, and a topology surrogate loss that penalizes topological errors by comparing persistence diagrams of prediction vs ground truth.

We concatenate the three components to form the combined adapter feature  $z^{H \oplus E \oplus S} = [z^H; z^E; z^S]$ , which has dimension  $d_H + d_E + d_S$ . This vector carries information encoded in three geometries. In practice, we set relatively small dimensions for each branch (e.g.  $d_H = d_E = d_S = 16$  or  $32$ ) to keep the adapter compact. The intuition is that  $z^H$  can encode hierarchical or coarse structural features (since distances in hyperbolic space accentuate global differences),  $z^S$  can encode boundary orientation or periodic patterns (e.g. the overall shape outline), and  $z^E$  retains fine details and any information not easily represented in the other manifolds.

During training, the hyperbolic and spherical outputs are treated with their respective distance metrics for certain loss terms (described below). However, when feeding into the decoder, we simply use the concatenated  $z^{H \oplus E \oplus S}$  as a unified feature vector for each spatial location. The adapter head contains the only new trainable weights in the feature extraction stage (three small linear projections). Freezing the main encoder  $\mathcal{F}$  means it continues to produce high-level features, while  $\mathcal{A}$  modulates and enriches those features in a task-specific way.

### 3.2 Segmentation Decoder

The decoder  $\mathcal{D}$  takes the adapter head’s output features and produces the final segmentation logits. We implement  $\mathcal{D}$  as a lightweight convolutional decoder inspired by DeepLabv3 ((1)). In our experiments, a simple design sufficed: we use a  $1 \times 1$  convolution to map the  $d_H + d_E + d_S$  dimensional adapter features to class logits for each spatial location (if the encoder output is lower resolution than the input image, we first upsample the feature map with bilinear interpolation or learned up-sampling). If the encoder provides multi-scale features (e.g. a FPN or ViT features from multiple layers), the decoder could also incorporate skip connections or concatenate features from different resolutions. In our implementation, for simplicity and efficiency, we focus on using the final layer features only and upsampling to the original image size by a factor  $\tau$  (e.g.  $\tau = 16$  for  $16\times$  down-sampled features).  $\mathcal{D}$  applies a few  $3 \times 3$  convolutions with ReLU to refine the segmentation map after upsampling.

Despite its simplicity, this decoder can produce accurate results when the encoder’s representations are strong. All decoder parameters are trained from scratch, but they constitute only a small fraction of the model (similar in size to the adapter head). Importantly, because the encoder is frozen, the decoder must learn to work with the given encoder features. Our added losses (below) assist in shaping these features for segmentation without modifying the encoder weights.

### 3.3 Hyperbolic InfoNCE Contrastive Loss

Adapting a frozen encoder with a small head can benefit from additional guidance beyond the supervised segmentation loss. We introduce a self-supervised contrastive objective applied to the hyperbolic branch of the adapter. The aim is to encourage the hyperbolic embeddings  $z^H$  to capture consistent global structure across perturbations of the input, and to cluster semantically similar regions closer together in the hyperbolic space.

We employ an InfoNCE-style loss ((10)) on pairs of embeddings extracted from two transformed versions of the same image. During training, for each input image  $I$ , we generate an augmented view  $I'$  (using random photometric and slight geometric augmentations such as color jitter or rotation). Both  $I$  and  $I'$  are passed through the frozen encoder and adapter head, yielding hyperbolic embeddings  $z^H$  and  $z'^H$  for each spatial position. We treat corresponding spatial locations in  $I$  and  $I'$  that depict the same part of the scene as positive pairs. For example, if the data comes with ground-truth segmentation, we can use matching class labels to identify positive pairs of pixels between  $I$  and  $I'$  (assuming the augmentation is mild so that spatial correspondence is roughly preserved). Alternatively, for completely unsupervised contrastive learning, one could use nearest-neighbor matching in feature space or assume the entire image representation is positive to itself.

We define the contrastive loss on the hyperbolic unit ball as:

$$L_{\text{hyp-InfoNCE}} = -\frac{1}{N_+} \sum_{(i,j) \in \text{pos}} \log \frac{\exp(-d_{\mathbb{B}}(z_i^H, z_j'^H)/\tau)}{\exp(-d_{\mathbb{B}}(z_i^H, z_j'^H)/\tau) + \sum_{(i,k) \in \text{neg}} \exp(-d_{\mathbb{B}}(z_i^H, z_k'^H)/\tau)}, \quad (1)$$

where  $d_{\mathbb{B}}(u, v)$  denotes the hyperbolic distance between two points  $u, v$  in the Poincaré ball,  $N_+$  is the number of positive pairs, and  $\tau$  is a temperature hyperparameter. In essence, for each positive pair  $(i, j)$  of corresponding features, the loss encourages their hyperbolic distance to be small, while pushing other non-matching features  $z_k'^H$  (negatives) away. We sample negatives from other spatial locations within the same image or from a batch of images.

Because distances in hyperbolic space grow quickly for points that are not similar (especially as they move toward the boundary of the Poincaré ball), this loss heavily penalizes points that should not be similar from clustering. Conversely, it allows features that belong to the same object or region to move closer (in a curved sense) in latent space. This process helps the adapter head  $z^H$  form more meaningful groupings, effectively learning a hierarchy: e.g., all pixels of an object might cluster around a certain hyperbolic point, different objects (even of the same class) might be farther apart reflecting a latent taxonomy (if the dataset has implicit hierarchies or instance distinctions).

Our hyperbolic InfoNCE loss is inspired by recent contrastive learning successes but is novel in its use on a hyperbolic manifold for structured segmentation features. It does not require additional memory beyond storing the features for the augmented pair, and we found it stabilizes training, preventing the small adapter from overfitting to just the supervised signal. It reinforces the notion

that the adapter’s hyperbolic space encodes global shape context, complementing the local focus of the Euclidean part.

### 3.4 Topology Surrogate Loss

While the above components guide the model to learn good features, the final segmentation still could suffer topological errors if optimized purely for pixel-wise accuracy. To directly address this, we integrate a topology-aware loss term  $L_{\text{topo}}$  that penalizes differences in topological features between the predicted mask and the ground truth. We leverage *persistent homology*, which provides a robust way to quantify topological features across all thresholds of a probability map ((4)).

Concretely, let  $\hat{Y}$  be the predicted probability map for the foreground (for binary segmentation,  $\hat{Y}(x) \in [0, 1]$  for each pixel  $x$ ; for multi-class, we can apply this to each object class separately or to the combined foreground mask). We compute its persistence diagram  $D(\hat{Y})$  for the sublevel set filtration: as the threshold  $\alpha$  goes from 0 to 1, we track the birth and death of topological features (connected components in  $H_0$  and loops/holes in  $H_1$ ) in the set  $\{x : \hat{Y}(x) \geq \alpha\}$ . Similarly, from the binary ground truth mask  $Y$ , we compute  $D(Y)$  which typically contains stable features corresponding to the actual topology (e.g., if the object should have one connected component and no holes,  $D(Y)$  will have a certain signature).

We then define  $L_{\text{topo}}$  by measuring the discrepancy between  $D(\hat{Y})$  and  $D(Y)$ . One common choice from prior work is the *Wasserstein distance* or *bottleneck distance* between diagrams ((2; 7)). We adopt a differentiable approximation similar to TopoLoss ((7)): we compute a soft matching between points in  $D(\hat{Y})$  and  $D(Y)$  and penalize unmatched or mis-matched points. Each topological feature is a point  $p = (b, d)$  in the diagram (birth and death threshold). If a feature in the prediction does not correspond to one in the truth, we effectively want to push its persistence ( $d - b$ ) to zero (i.e., remove that feature), and if a feature exists in truth but not in prediction, we want to create it.

We implement this by first matching each point  $p \in D(Y)$  to the closest point  $q \in D(\hat{Y})$  in terms of diagram distance (with the convention that points can also match to the diagonal if no actual feature is present, representing “noise”). Let  $\|p - q\|$  be the  $L_2$  distance in the diagram plane. Our topology loss can be written as:

$$L_{\text{topo}} = \sum_{p \in D(Y)} \min_{q \in D(\hat{Y}) \cup \{\text{diag}\}} \|p - q\|^2 + \sum_{q \in D(\hat{Y})} \min_{p \in D(Y) \cup \{\text{diag}\}} \|p - q\|^2.$$

Here the first term sums the squared distance for each true feature  $p$  to the nearest predicted feature  $q$  (or to diagonal if unmatched), and the second term does the same from predicted features to true ones. Intuitively, this penalizes any extra or missing features proportionally to their significance (distance from diagonal, which is persistence). If the predicted mask has an extra hole that ground truth doesn’t,  $D(\hat{Y})$  will have a point for that hole far from diagonal, while  $D(Y)$  might not; the loss will push that point toward the diagonal (reducing the hole’s persistence, effectively filling the hole). Conversely, if the ground truth has a hole and the prediction doesn’t, the loss will push the prediction to create one.

Because computing persistent homology on every training iteration is expensive, we compute  $D(\hat{Y})$  and  $D(Y)$  on a subsampled batch of predictions (or after a certain number of epochs) and use interpolation to estimate gradients for  $\hat{Y}$ . We use the GUDHI library ((15)) to compute diagrams. GUDHI provides persistence pairs and we differentiate through the process by treating the birth/death values as functions of the probability map. In practice, we approximate the gradient by considering small perturbations to  $\hat{Y}$  and their effect on  $D(\hat{Y})$ . Although this surrogate gradient is not exact, prior work ((7)) has demonstrated that it suffices to meaningfully reduce topological errors.

We find that combining  $L_{\text{topo}}$  with standard losses effectively guides the model to prefer predictions with correct topology when possible. Notably, this does not significantly harm pixel accuracy; rather, it often acts as a regularizer that prevents odd mistakes (like a segmented structure breaking apart into two pieces to chase a slight pixel-wise gain). Figure 2 illustrates a scenario where a model without topological loss produces a nearly correct mask but fills in a critical hole. With the topology surrogate, the model learns to preserve that hole, aligning with the true anatomy.

figures/topology\_surrogate\_diagram.pdf

Figure 2: **Topology Surrogate Illustration.** We visualize the persistent homology computation on a sample prediction vs ground truth. (a) Ground truth mask with one connected component and one hole (Betti  $(\beta_0 = 1, \beta_1 = 1)$ ). (b) Prediction misses the hole (topologically incorrect  $\beta_1 = 0$ ) despite similar Dice. (c) Persistence diagrams: ground truth has a prominent  $H_1$  feature (red point) off the diagonal, whereas prediction’s  $H_1$  feature is either absent or close to diagonal (tiny persistence). Our topology loss penalizes this discrepancy, pushing the model to carve out the missing hole in the prediction.

### 3.5 Overall Training Objective

The total loss for training HyperTopo-Adapters is a weighted sum of the segmentation loss, hyperbolic InfoNCE, and topology surrogate:

$$L_{\text{total}} = L_{\text{seg}} + \lambda_c L_{\text{hyp-InfoNCE}} + \lambda_t L_{\text{topo}},$$

where  $L_{\text{seg}}$  is typically the cross-entropy or Dice loss for segmentation accuracy,  $\lambda_c$  and  $\lambda_t$  are hyperparameters controlling the importance of contrastive and topological terms. In our experiments, we set these  $\lambda$  values such that the additional losses are on the same order of magnitude as  $L_{\text{seg}}$  at the start of training (e.g.  $\lambda_c = 0.1$ ,  $\lambda_t = 0.01$  in some cases). We found that the training is not overly sensitive to small changes in these weights; the topological loss especially can be quite low-weight (since its signals are sparse yet important).

All components of  $L_{\text{total}}$  are differentiable or sub-differentiable with respect to the adapter and decoder parameters. The frozen encoder has no trainable parameters, and thus we do not backpropagate into it. Our approach thereby retains the efficiency benefits of not updating the large backbone (which could have millions or billions of parameters), focusing computation on the relatively tiny adapter and decoder.

## 4 Experiments

We evaluate HyperTopo-Adapters on several image segmentation tasks, comparing against both traditional full-finetuning models and other adapter-based baselines. We aim to answer: (1) Does our method achieve competitive segmentation accuracy (Dice/IoU) with significantly fewer trainable parameters? (2) Does it produce segmentations with more correct topology (fewer splits, holes) as intended? (3) How do the geometric and topological components each contribute to performance?

**Datasets.** We conduct experiments on two representative datasets. First, a **Synthetic Shapes** dataset designed to test topological generalization: it consists of 2D binary images containing one or two simple shapes (circles, donuts, etc.) with known topology (e.g. a donut shape has exactly one hole). There are 5,000 training and 1,000 test images, with ground truth masks where we know the true Betti numbers. This dataset evaluates whether models can learn to preserve topology under varying shape sizes and positions. Second, a **Medical MRI** dataset derived from cardiac MRI scans (based on the public ACDC dataset), where the task is to segment the left ventricular cavity, myocardium, and right ventricle. These structures have well-defined topology (each should be a single connected component and cavity has a hole inside myocardium). We use 100 short-axis MRI slices for training and 50 for testing, with manual annotations. This dataset is small and challenging, making it ideal for adapter-based training rather than full network training.

**Baselines.** We compare our approach to several baselines: **(a) U-Net (scratch):** a U-Net ((13)) trained from scratch on the target data (full finetuning of all weights). **(b) DeepLabv3 (ResNet-50):** DeepLabv3 ((1)) using a ResNet-50 ((5)) backbone initialized from ImageNet, then finetuned on the segmentation task (typical transfer learning). **(c) Full Finetune DINO:** a ViT encoder pretrained with DINOv2 ((11)), finetuned end-to-end (all transformer weights updated) with a segmentation head (we use a small conv decoder similar to ours for fairness). **(d) Frozen DINO + Linear:** a frozen DINOv2 encoder with only a linear classifier or convolution tuned on top (no special losses, no manifold adapter). This represents a naive adapter without our contributions. **(e) Frozen DINO + Basic Adapter:** a stronger baseline where we attach a small MLP adapter (Euclidean only) similar in parameter count to our  $H \oplus E \oplus S$  head, and train with just cross-entropy.

Our model **(f) HyperTopo-Adapter (Ours)** uses a frozen DINOv2 backbone with our manifold head, hyperbolic contrastive loss, and topology loss. For a fair comparison, all methods use the same training/validation splits and evaluation protocol.

**Metrics.** We report standard overlap metrics: **Dice coefficient** ((3)) (F1-score of overlap) and **Intersection-over-Union (IoU)** ((8)) for each foreground class, as well as the average across classes. To specifically evaluate topology, we compute two metrics: **Boundary F1 score (BF)** ((12)) which measures how well the predicted boundary aligns with the ground truth boundary (tolerance of 1–2 pixels, reported as an F1 score), and **Betti error count**, which counts the absolute difference in number of connected components and holes between prediction and truth (lower is better). BF is a proxy for topological correctness because a high BF indicates that the shape outline is captured correctly. All metrics are computed on the test set for final models. We select model checkpoints based on validation Dice score unless otherwise noted.

**Implementation.** Our backbone is the ViT-B/16 architecture from DINOv2 ((11)), which yields a  $14 \times 14$  feature grid for a  $224 \times 224$  input (patch size 16). The adapter head uses  $d_H = d_E = d_S = 16$ . The decoder upsamples by  $16 \times$  with a 4-layer conv network. We train using Adam optimizer, batch size 8, for 200 epochs on synthetic and 50 epochs on MRI (the dataset is small). The initial learning rate is  $1e-3$  for adapter/decoder and we use cosine decay.  $\lambda_c = 0.1, \lambda_t = 0.01$  for synthetic, and slightly lower  $\lambda_t = 0.005$  for MRI (since the labels can be slightly noisy). Data augmentation includes random flips, rotations (up to 10 degrees), and intensity shifts. All experiments run on a single NVIDIA A100 GPU; training our model takes 1 hour for synthetic and 30 minutes for MRI, which is 3–4 $\times$  faster than finetuning the entire DINO backbone.

## 5 Results

We present quantitative and qualitative results, followed by ablation studies.

**Quantitative Performance.** Table 1 summarizes the main results on both datasets. On the **Synthetic Shapes**, our HyperTopo-Adapters achieve a Dice of 0.981 and IoU of 0.964, essentially matching the best full-finetuning baseline (DeepLabv3) in overlap accuracy. Notably, we significantly outperform others on Boundary F1 (0.95 vs 0.87 for next best) and have zero topological errors in all test images (the only method to do so). The U-Net and DeepLab baselines, while high in Dice, often produced small spurious regions or missed holes in a few cases, giving a non-zero Betti error count. The frozen DINO + linear head baseline struggled on this task (Dice 0.90) since it cannot capture the shape complexities with just a linear classifier on fixed features. Adding our adapter

Table 1: Segmentation performance on Synthetic Shapes and Cardiac MRI. We report Dice (higher better), IoU, Boundary F1 (BF), and topology errors (Betti Err). Best results in **bold**. HyperTopo-Adapters matches or exceeds full models in overlap scores while greatly reducing topological errors.

Method	Synthetic Shapes (binary)				Cardiac MRI (multi-class)			
	Dice $\uparrow$	IoU $\uparrow$	BF1 $\uparrow$	Betti Err $\downarrow$	Dice (mean) $\uparrow$	IoU (mean) $\uparrow$	BF1 $\uparrow$	Betti Err $\downarrow$
(a) U-Net (scratch)	0.972	0.947	0.865	0.15	0.872	0.801	0.754	1.2
(b) DeepLabv3 (Res50)	0.984	0.969	0.882	0.10	0.895	0.825	0.773	0.8
(c) Full Finetune DINO	<b>0.987</b>	<b>0.973</b>	0.902	0.08	0.908	0.834	0.781	0.5
(d) Frozen DINO + linear	0.901	0.820	0.712	0.60	0.830	0.740	0.720	2.3
(e) Frozen DINO + basic adapter	0.948	0.900	0.805	0.35	0.881	0.804	0.760	1.5
<b>(f) Ours: HyperTopo-Adapter</b>	0.981	0.964	<b>0.950</b>	<b>0.00</b>	<b>0.901</b>	<b>0.832</b>	<b>0.840</b>	<b>0.0</b>

without special losses (baseline e) improved Dice to 0.95 but still had many topological mistakes (Betti error in 18% of test samples). Our full model (f) eliminates those mistakes, demonstrating the value of the topology loss. On the **Cardiac MRI** dataset, our method again attains competitive Dice/IoU (e.g. mean Dice 0.901 vs 0.908 for full finetune DINO, essentially equivalent within margin) while yielding significantly better BF (0.84 vs 0.78) and fewer connectivity errors. For instance, segmentations from the baseline often had the myocardium split into two pieces or a missing cavity in some slices, whereas ours correctly preserved one connected myocardial ring with a cavity. The improvements in BF1 indicate cleaner, more accurate boundaries—an important factor for clinical use where delineating exact organ shape is critical. Importantly, these gains come with only ~5% of the trainable parameters of the full model. This highlights that our geometry+topology-guided adapter makes the most of the frozen encoder’s features.

The results confirm that our method does not sacrifice accuracy for topology: in fact, on Cardiac MRI, our Dice is slightly higher than U-Net and nearly equals full DINO finetuning, indicating that the additional constraints did not hinder learning. The boundary F1 boost of 6-7 points on MRI implies that edges of structures are more precisely segmented (likely due to the spherical embedding capturing boundary orientations plus the topological smoothing effect).

**Qualitative Results.** Figure 3 provides a visual comparison of segmentation outputs. In the Synthetic Shapes examples (top row), U-Net and DeepLab produce mostly correct masks but occasionally fill in a hole or break a shape—see panel (b) where U-Net output missed the hole in a donut shape (arrow). Our HyperTopo-Adapter output (c) correctly identifies the donut hole. In panel (e), a case with two touching shapes, the baseline misclassified them as one blob whereas ours delineated the boundary between them properly (yielding two components as ground truth). For the MRI examples (bottom row), our model’s output (h) shows a smooth, continuous myocardium and a clear cavity, closely matching the ground truth (g). In contrast, the baseline (DeepLab) in (f) fragmented the myocardium (note the gap in the wall, arrow) and had an irregular boundary. These qualitative differences are subtle but important in practice (e.g., a broken myocardial wall would indicate an error in volumetric quantification). The HyperTopo-Adapter outputs consistently appear more regular and closer in topology to the true masks across many samples.

**Topology vs Accuracy Trade-off.** An interesting question is whether optimizing for topology harms Dice accuracy, or if both can be improved simultaneously. To investigate this, we trained a series of models with varying  $\lambda_t$  (strength of topology loss) and plot the resulting Dice score vs persistent homology discrepancy (we use average bottleneck distance as a measure) on the validation set. Figure 4 illustrates this relationship. Without any topology loss ( $\lambda_t = 0$ ), the model achieves slightly higher Dice initially but with a large persistent diagram distance (i.e., topology very off). As  $\lambda_t$  increases, the persistent homology error drops dramatically, and Dice only decreases marginally (in some ranges, Dice even increases due to the regularization effect). We choose a moderate  $\lambda_t$  that gives the best balance (highlighted point). This shows that with our method, we can improve topology a great deal with only a minor trade-off in pixel accuracy, and in some cases no trade-off at all. In fact, our final model (with topology loss) ended up with slightly *better* Dice on the MRI set than the one without, likely because enforcing the correct structure prevented some overfitting to noisy labels. This result dispels the notion that one must sacrifice pixel accuracy to enforce shape priors; with a proper surrogate loss and strong encoder features, the model can satisfy both criteria.

figures/segmentation\_overlay\_gallery.pdf

Figure 3: **Segmentation Examples.** Top row: Synthetic shapes. (a) Input image. (b) U-Net result (red outline) vs ground truth (green); the U-Net incorrectly filled the donut hole (arrow). (c) Our result (blue outline) vs ground truth (green); the hole is correctly preserved. Bottom row: Cardiac MRI. (d) Input slice. (e) DeepLabv3 baseline output (red) vs GT (green); baseline segmentation has a discontinuity in the heart wall (arrow) and some jagged edges. (f) Our output (blue) vs GT; it yields a single connected ring for the myocardium and smooth boundaries, matching the expected anatomy.

**Ablation Studies.** We perform extensive ablations to justify each component of HyperTopo-Adapters. Table 2 shows the results on the cardiac MRI validation set when we remove or modify one component at a time: (1) *No Hyperbolic* – using only  $E \oplus S$  (Euclidean + spherical) in the adapter, (2) *No Spherical* – using  $H \oplus E$  only, (3) *Euclidean-only head* – a single Euclidean MLP (similar parameter count), (4) *No InfoNCE* – removing the hyperbolic contrastive loss, (5) *No Topo Loss* – removing  $L_{\text{topo}}$ , and (6) *Full Finetune Encoder* – we unfreeze the encoder and train all weights (to see if freezing hurts performance).

From the ablations: the manifold head clearly provides benefits – using only Euclidean features (row 3) drops Dice by about 2 points and BF by 4 points, indicating that hyperbolic and spherical components did contribute to better segmentation and boundary adherence. Removing the hyperbolic part (row 1) hurt performance more than removing spherical (row 2), suggesting the hyperbolic embedding is particularly useful, likely due to the strong shape-context it provides. The spherical part has a smaller effect, but still improves BF1 slightly (makes sense, as spherical coordinates might help

figures/dice\_vs\_pd\_plot.pdf

Figure 4: **Dice vs Topology Trade-off.** Validation performance as we vary the weight of the topology surrogate loss. The x-axis is the average bottleneck distance between predicted and true persistence diagrams (lower means better topological alignment), and y-axis is the Dice score. Our full model (star) achieves a good compromise, markedly improving topology (leftward shift) with virtually no loss in Dice. In fact, moderate topology regularization can slightly boost Dice by eliminating anatomically implausible predictions.

Table 2: Ablation study on validation set (Cardiac MRI). We remove one component at a time to measure its effect.

Model Variant	Dice $\uparrow$	IoU $\uparrow$	BF1 $\uparrow$	Betti Err $\downarrow$
Full HyperTopo-Adapter (H+E+S, w/ InfoNCE, w/ Topo)	0.902	0.829	0.835	0.1
(1) No Hyperbolic (E+S only head)	0.890	0.816	0.823	0.3
(2) No Spherical (H+E only head)	0.896	0.822	0.828	0.2
(3) Euclidean-only head	0.883	0.808	0.795	0.4
(4) No Hyperbolic InfoNCE loss	0.894	0.818	0.814	0.4
(5) No Topology surrogate loss	0.907	0.832	0.789	0.7
(6) Unfreeze encoder (finetune backbone)	0.903	0.830	0.823	0.3

fine-tune boundaries). The hyperbolic InfoNCE loss (row 4) also has a noticeable effect: without it, Dice drops and the topological error increases a bit – the model likely overfits more to pixel-wise loss without the contrastive regularization ensuring feature consistency. The topology surrogate (row 5) had the largest impact on topology metrics: without it, Betti errors jump from near 0 to 0.7 on average (meaning many images have a wrong number of components/holes), and BF1 declines. Interestingly, without topology loss, Dice improved marginally (from 0.902 to 0.907), aligning with the earlier observation that there is a tiny trade-off if we don’t weigh it carefully. But we argue the structural correctness is more valuable, and in our final model the difference in Dice is negligible. Finally, unfreezing the encoder (row 6) did not yield a significant boost in Dice (0.903 vs 0.902) but slightly degraded BF1 and topology (likely due to over-training on limited data). This validates our choice to keep the heavy encoder fixed – it not only saves computation but also avoids overfitting, while our adapter can still reach high performance.

These ablations confirm that each proposed component contributes meaningfully: the  $H \oplus E \oplus S$  design yields richer features than any single space alone; the hyperbolic contrastive loss improves feature quality and stability; and the topology loss is indispensable for enforcing structural correctness. The fact that our method works so well with the encoder frozen (and even performs a bit worse when unfreezing) underscores the effectiveness of guiding a strong pretrained model with the right inductive biases instead of brute-force finetuning.

## 6 Conclusion

We presented HyperTopo-Adapters, a novel approach to segmentation that injects geometric and topological intelligence into the adaptation of frozen encoders. By leveraging hyperbolic and spherical embeddings, we enhanced the representation of a pretrained vision transformer to capture global and boundary information. The integration of a persistent-homology-based loss allowed us to explicitly steer the segmentation outputs toward correct topological structures. Our experiments demonstrated that with a fraction of the trainable parameters, HyperTopo-Adapters can match the accuracy of fully finetuned models while delivering substantially better topological fidelity.

This work opens several avenues for future research. One direction is to extend the method to multi-class and instance segmentation in more complex scenes—handling interactions between object topologies (for instance, preserving nesting relationships). Another direction is exploring other geometric spaces or learned manifolds in the adapter head (e.g., product of multiple hyperbolic spaces to capture multiple hierarchies). Additionally, scaling the approach to 3D volumetric segmentation (where topology in 3D is even more interesting) would be valuable for medical imaging. Finally, while we focused on frozen foundation models, the idea of topology-aware losses can be combined with partial finetuning in cases where some encoder layers are adapted, potentially yielding the best of both worlds.

In summary, our work demonstrates a practical way to achieve geometry- and topology-aware segmentation by combining modern representation learning with classical topological priors. We hope that HyperTopo-Adapters inspire further integration of topological data analysis techniques in deep learning pipelines, especially as the field moves toward ever larger pretrained models that need task-specific grounding. Ensuring that neural network outputs respect fundamental structure is an important step toward reliable and interpretable AI in vision.

## A Additional Metrics and Loss Details

In this appendix section, we provide precise definitions for the evaluation metrics used in the main paper and expand on certain loss function formulations.

### Dice, IoU, and Other Evaluation Metrics

The **Dice coefficient** (Sørensen–Dice index) we report for a given class is defined as

$$\text{Dice} = \frac{2|P \cap T|}{|P| + |T|},$$

where  $P$  is the set of predicted foreground pixels and  $T$  is the set of ground-truth foreground pixels (for that class) ((3)). It ranges from 0 to 1, with 1 indicating a perfect segmentation overlap. We often multiply by 100 to express it as a percentage. Intersection-over-Union (**IoU**, a.k.a. the Jaccard index) is

$$\text{IoU} = \frac{|P \cap T|}{|P \cup T|},$$

the ratio of overlap area to the union of predicted and true areas ((8)). IoU is generally more strict than Dice; for reference,  $\text{Dice} = \frac{2 \cdot \text{IoU}}{1 + \text{IoU}}$ . We report mean IoU across classes for multi-class segmentation.

The **Boundary F1 (BF) score** ((12)) is an F1-score computed on the boundary pixels of the objects. We follow the protocol from Perazzi *et al.* ((12)): the boundary of the predicted mask is compared to the boundary of the ground truth, counting a true positive if a predicted boundary pixel lies within 2 pixels of any ground-truth boundary pixel (and vice versa for recall). Precision and recall of boundary detection are computed, then  $\text{BF} = 2 \cdot (\text{prec} \cdot \text{rec}) / (\text{prec} + \text{rec})$ . BF1 offers insight into the quality of the shape outline irrespective of interior overlap.

In addition to these, for analysis we also looked at the **Hausdorff distance** (95th percentile) between the predicted and true boundaries, which quantifies the worst-case deviation. We found that our method reduced the 95% Hausdorff by 20% compared to baselines on the MRI set, correlating with the improved BF score (though we did not include Hausdorff in the main paper due to space).

## Loss Function Details

The main text described the hyperbolic InfoNCE and topology surrogate qualitatively. Here we add further details or alternatives:

For the **hyperbolic InfoNCE** loss, one important detail is how positives and negatives are defined. In our supervised contrastive variant, we considered as positives all pixel embeddings within the same ground-truth region (for an image and its augmented view). Alternatively, one can pick one anchor pixel and define its positive as the corresponding pixel in the augmented image (if augmentation is mostly rigid), or a small neighborhood around that location. We experimented with both and found that using the entire region as positive set gave a stronger signal: effectively, all pixels of the same object (or class) are pulled together, which complements the cross-entropy loss that treats each pixel independently. The temperature  $\tau$  was set to 0.07 in our experiments, a common value in contrastive learning. We also used  $L_2$  normalization on the hyperbolic embeddings before computing distances to improve numerical stability (distance in the Poincaré ball can be large, so we normalize the ball radius to 1 and ensure embeddings aren’t too close to the boundary initially).

For the **topology surrogate loss**, one key implementation note is that we used binary masks thresholded at 0.5 for computing persistence diagrams in practice. That is, we didn’t compute full sublevel filtrations at every training step (which would be costly). Instead, at each iteration we compare the topology of the current hard prediction (mask after thresholding at 0.5) to the ground truth mask. This reduces the problem to comparing two binary shapes. In this case, persistent homology of a binary image is trivial in that each feature is either present or absent (we essentially detect connected components and holes using standard algorithms, and penalize discrepancies). This is a simpler surrogate than using the full continuous-valued PH as Hu *et al.* ((7)) did, but we found it effective and much faster. Because our segmentation outputs are fairly high-confidence (thanks to the strong encoder), thresholding at 0.5 did not lose much information. In future work, one could incorporate a more advanced PH computation that takes into account the confidence of predictions (like ensuring a hole with probability 0.9 is penalized more than one with probability 0.4).

We also note that we tried a variant of  $L_{\text{topo}}$  that directly targets Betti numbers: e.g.,

$$L_{\beta} = (\hat{\beta}_0 - \beta_0^{gt})^2 + (\hat{\beta}_1 - \beta_1^{gt})^2,$$

where  $\hat{\beta}_i$  are the predicted  $i$ th Betti numbers (number of components and holes) and  $\beta_i^{gt}$  are ground truth. This loss is even simpler (just matching counts) but it doesn’t tell the model *where* to fix the topology. We observed that  $L_{\beta}$  alone was not sufficient—the model might remove a hole but in a random manner, or merge components incorrectly just to get the count right. In contrast, the persistence diagram approach penalizes the size of features, encouraging minimal modifications to fix topology (like closing the smallest hole, or connecting the nearest components). Thus, our chosen loss provided more guidance than a naive Betti-matching loss.

Finally, our **segmentation loss**  $L_{\text{seg}}$  was a combination of soft Dice loss and cross-entropy. For each class, we computed Dice loss  $L_{\text{Dice}} = 1 - \text{Dice}$  (differentiable approximation using probabilities) and added it to the standard cross-entropy. This helped in class imbalance situations (the ventricle cavity is small relative to background, for instance). The weight for Dice loss was 1.0 and for cross-entropy was 1.0 (equal weighting) in our final models.

## B Topology Surrogate and Persistent Homology Computation

Here we provide additional background on how we computed persistent homology and how the topology surrogate loss was implemented efficiently.

### Persistent Homology Background

Persistent homology (PH) is a tool from computational topology that analyzes the multi-scale topological features of shapes ((4)). In an image context, one typically considers a filtration of a binary image as the threshold varies. For an image function  $f(x)$  giving (say) foreground probability at pixel  $x$ , one defines sublevel sets  $S(\alpha) = \{x : f(x) \geq \alpha\}$  for  $\alpha \in [0, 1]$ . As  $\alpha$  goes from 1 down to 0,  $S(\alpha)$  transitions from empty to full image, and topological features (components and holes) appear and eventually merge or fill in. PH tracks these via birth and death  $\alpha$  values, producing a persistence diagram: each feature is a point  $(\alpha_{\text{birth}}, \alpha_{\text{death}})$ .

We were primarily concerned with  $H_0$  (components) and  $H_1$  (holes) features in 2D. The ground truth binary mask can be seen as  $S(\alpha)$  at  $\alpha = 0.5$  (assuming a crisp mask). If the prediction were also binary, differences in topology are easy to find by standard algorithms (like connected component labeling for  $H_0$  and hole finding via connected components of the complement for  $H_1$ ). For grayscale predictions, one could use a library like GUDHI ((15)) to compute PH. GUDHI requires constructing a simplicial complex (often one uses the cubical complex for images, where each pixel and its 8-neighbor connectivity define topology). We used GUDHI’s cubical complex module for some experiments, but as noted, doing so every iteration is slow for large images.

One trick we used was to restrict topology checks to the largest features: we only computed PH on a downsampled version of the mask or within bounding boxes around objects to save time. Since our topology focus was mainly on presence/absence of holes and splits, even a coarse PH could catch those differences.

### Efficient Implementation of Topology Loss

During training, we implemented  $L_{\text{topo}}$  as follows:

1. Threshold predicted mask  $\hat{Y}$  at 0.5 to get binary  $\hat{B}$ .
2. Compute connected components of  $\hat{B}$  (using union-find or flood-fill). Compute connected components of ground truth  $B^{gt}$ .
3. If number of components differ, identify smallest predicted components that could be merged or extraneous ones to remove. We add a penalty proportional to the area of those extraneous components (this approximates pushing them to disappear).
4. Compute holes: we filled  $\hat{B}$  and  $B^{gt}$  and looked at connected components of the background to find holes. If  $\hat{B}$  has a hole that  $B^{gt}$  doesn’t, we add penalty proportional to area of that hole (encouraging it to fill). If  $B^{gt}$  has a hole and  $\hat{B}$  doesn’t, we add penalty proportional to that hole area (encouraging the model to carve it out).

This approach essentially implements a simplified persistence matching: large extraneous holes/components incur larger loss. We used morphological operations (open/close) to guide where to add/remove regions in  $\hat{B}$ .

We also ensured differentiability by not stopping gradient at the threshold. Instead of a hard 0.5 threshold, we used a smooth approximation:  $H_\epsilon(x) = \frac{1}{2}(1 + \tanh(\frac{x-0.5}{\epsilon}))$  as a soft Heaviside step (with  $\epsilon$  small, e.g. 0.1). This allowed gradients to flow through the binary map computation. The rest of the operations (component detection, etc.) were treated as piecewise constant for gradients—we did not compute exact gradients for those, but we backpropagated the loss on pixel sets (for example, if an extra component of mask corresponds to a set of pixels  $S$ , we add a loss  $\lambda \sum_{x \in S} \hat{Y}(x)$  which will encourage those pixels’ probabilities to drop). This way, we directly nudge the probabilities of offending regions down, or of missing regions up (by a complementary term encouraging some pixels to increase if a hole is needed). These heuristic gradient assignments were effective in practice to reduce topology errors after a few training iterations.

### Validation of Topology Surrogate

We verified that our surrogate indeed correlates with true topological differences by testing on known shapes. For example, if ground truth is a ring shape (1 hole) and prediction lacks the hole, our loss was high and its gradient correctly pointed towards lowering the center region’s prediction (thus creating a hole). After training, that term went to near zero as the hole opened up. This indicates the surrogate provided the correct signal. It might not strictly minimize the exact bottleneck distance, but it was sufficient to achieve the desired outcome (matching topology).

## C Configuration and Training Regime

We detail the model configurations and training settings for reproducibility.

## Model Architecture Configurations

- **Encoder:** ViT-Base (ViT-B/16) from DINOv2. 12 transformer layers, 768-dim embeddings, patch size 16, image input  $224 \times 224$ . We used the official DINOv2 weights (pre-trained on a large dataset) without modification.
- **Adapter Head:** Three linear layers (fully connected) of sizes:  $768 \rightarrow 16$  (hyperbolic),  $768 \rightarrow 16$  (Euclidean),  $768 \rightarrow 16$  (spherical). The hyperbolic and spherical had additional non-linearities: hyperbolic branch applied  $\tanh$  to each output and then scaled by factor 0.95 to ensure staying inside the Poincaré ball radius (just a precaution), spherical branch applied an  $L2$  norm. These layers were initialized with small random weights ( $\mathcal{N}(0, 0.01)$ ) and zero biases.
- **Decoder:** For Synthetic Shapes (binary), a single  $1 \times 1$  conv from 48 channels ( $16+16+16$ ) to 1 output channel was used, followed by bilinear upsampling to original size. For MRI (3 classes + background), we used a slightly larger decoder: 48-channel features input  $\rightarrow$  two  $3 \times 3$  conv (ReLU)  $\rightarrow$  16-channel feature map  $\rightarrow$  upsample  $\times 2$  (nearest + conv refine) repeatedly until original size (in our case,  $14 \times 14$  to  $224 \times 224$  is  $16 \times$  upsampling, which we did in four  $\times 2$  steps). Skip connections from the adapter output (at lower res) were concatenated during upsampling steps to preserve detail. The final layer was a  $1 \times 1$  conv to 3 logits (for three classes) plus background implicitly.

Total trainable parameters: Adapter head ( $768 \times 16 \times 3 = 36.8k$  + biases), Decoder (small conv network, 50k params). In total roughly 0.09 million. The ViT-B encoder has 86 million which remained frozen. By comparison, finetuning the whole ViT would involve all 86M.

## Training Hyperparameters

- Optimizer: AdamW (weight decay  $1e-4$  for adapter/decoder weights). We did not regularize the backbone since it's frozen.
- Learning Rate Schedule: Cosine decay from initial LR to  $1e-6$ . Initial LR for adapter/decoder was  $1e-3$  (Synthetic) and  $5e-4$  (MRI, due to smaller data).
- Batch Size: 16 for Synthetic, 8 for MRI (due to higher memory usage per image).
- Augmentations: For Synthetic, random rotation (0-15 degrees), random shape scaling (the shapes themselves were procedurally generated differently each epoch), and salt-and-pepper noise. For MRI, random horizontal flip, small random rotation (0-5 degrees), random intensity scaling (brightness/contrast jitter).
- Contrastive augmentation: We generated the augmented view  $I'$  on-the-fly for InfoNCE by applying an additional random color jitter and Gaussian blur on top of the base augmentation. We found having at least slight differences (like blur) helps the model not trivially match pixels.
- Loss weights: as stated,  $\lambda_c = 0.1$  for Synthetic (since there we had no labels, it was purely unsupervised InfoNCE, we needed a bit stronger weight to keep features aligned), and 0.05 for MRI (where supervised InfoNCE used labels, a smaller weight sufficed).  $\lambda_t = 0.01$  for Synthetic, 0.005 for MRI. We did a grid search over  $\{0.1, 0.01, 0.001\}$  and those gave the best topology without hurting Dice.
- Gradient clipping: we clipped gradients norm at 1.0 to avoid any explosion (especially due to topology loss which can spike early on).
- Early stopping: not used formally, but we observed that after around 150 epochs (Synthetic) and 40 epochs (MRI) the validation metrics converged. We simply trained to the full budget (200/50) and picked the last model (which in these cases was also the best).

We also implemented a simple learning rate warmup of 5 epochs at the start (linearly increasing from 0 to initial LR) to stabilize training, given the contrastive loss can cause instability if started too high.

## Computing Environment

Experiments were run on Python 3.10 with PyTorch 1.13. GPU: NVIDIA A100 40GB. The persistent homology computations were done on CPU using GUDHI 3.5.0 (for any heavy batch we

offloaded to CPU to avoid blocking GPU). The code will be released with careful documentation to reproduce all results.

### Additional Observations

During training, we noticed the contrastive loss in hyperbolic space led to very tight clustering of features for the same object. We occasionally monitored the distribution of hyperbolic distances: initially, they were around the same scale as Euclidean distances, but as training progressed, points belonging to the same region collapsed closer (distance  $\sim 0.2$  in Poincaré metric) while different regions were far apart ( $> 1.0$ ). This separation is much more pronounced than in Euclidean feature space. We interpret this as the hyperbolic space emphasizing the between-object differences (because to accommodate many points, it pushes them towards the boundary, effectively amplifying separation). This was beneficial for segmentation, as pixels from different objects became easier to classify by a linear separator in the decoder.

Another observation: the topology loss initially spiked when the model output was nonsense (random). But within a few epochs, the model learned to avoid obvious topological errors because those gave large loss. For example, in early iterations, a prediction might be just random noise (dozens of components), yielding huge  $L_{\text{topo}}$  compared to a single component ground truth. The model quickly adjusted to output one component to reduce that loss, even before getting the shape right. This indicates the topo loss provided a strong coarse correction (like “ensure one connected blob”). Then the segmentation loss fine-tuned the exact boundaries of that blob.

In summary, the training regime successfully balances pixel accuracy and structural accuracy, and the chosen hyperparameters should transfer to similar settings (we expect they might need scaling if using a different backbone or dataset, e.g. a larger dataset might require smaller  $\lambda_t$  since the model has more data to learn topology from implicitly).

### References

- [1] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [2] James R. Clough, Nicholas Byrne, Ilkay Oksuz, Veronika A. Zimmer, Julia A. Schnabel, and Andrew P. King. A topological loss function for deep-learning based image segmentation using persistent homology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(1): 1555–1566, 2020.
- [3] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.
- [4] Herbert Edelsbrunner and John Harer. Persistent homology – a survey. *Surveys on Discrete and Computational Geometry: Twenty Years Later*, 453:257–282, 2008.
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [6] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [7] Xiaoling Hu, Li Fuxin, Dimitris Samaras, and Chao Chen. Topology-preserving deep image segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 7073–7085, 2019.
- [8] Paul Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 11(2):37–50, 1912.
- [9] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pages 6338–6347, 2017.

- [10] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [11] Maxime Oquab, Timoth’ee Darcet, Th’eo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
- [12] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–732, 2016.
- [13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.
- [14] Oriane Sim’eon, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Micha’el Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timoth’ee Darcet, Th’eo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Herv’e J’egou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.
- [15] The GUDHI Project. *GUDHI User and Reference Manual*, 2015. GUDHI Editorial Board.