FILTR: Extracting Topological Features from Pretrained 3D Models

Supplementary Material

We provide implementation details, including the creation of DONUT (Sec. 6.1) and the architecture of FILTR and baselines (Sec. 6.3) along with the training procedure. We also present additional experimental results for probing (Sec. 8.2) and feature alignment (Sec. 8.3), as well as additional experiments to further motivate design choices for FILTR (Sec. 8.4). Finally, we include qualitative results on persistence diagram prediction. All the code and data to reproduce our experiments are available at https://filtr-topology.github.io/.

6. Implementation details

6.1. Creation of DONUT

The primary goal in constructing DONUT is to obtain reliable and balanced topological annotations. The generation pipeline (Fig. 9) therefore first samples valid global labels, then distributes them across components, and finally produces geometrically diverse meshes consistent with the prescribed topology.

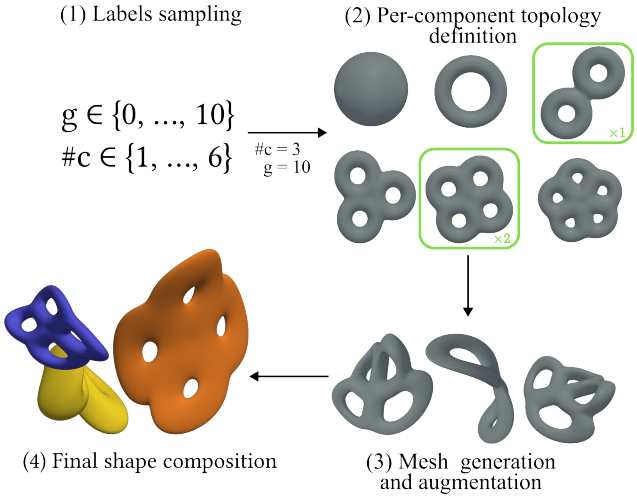

Figure 9. **DONUT generation pipeline.** (1) Sample global topological labels (Alg. 1); (2) distribute them across components (Sec. 6.1.1); (3) generate each component mesh (Sec. 6.1.2); (4) apply component-wise augmentations and merge them without overlap to preserve global topology.

6.1.1. Labels sampling

Label generation is performed prior to mesh construction and is controlled by a small set of hyperparameters. For each sample, we draw its number of connected components and total genus under the following constraints:

- The total genus does not exceed $G^{\max}$.
- The genus of each component does not exceed $g^{\max}$.
- The number of connected components lies in $[\![\beta_0^{\min}, \beta_0^{\max}]\!]$.
- The marginal distribution of labels is approximately uniform.

Hyperparameter	Value
$g^{\max}$	5
$G^{\max}$	10
$\beta_0^{\min}$	1
$\beta_0^{\max}$	6
k	1000

Table 4. Hyperparameter values used to create DONUT.

Algorithm 1 summarizes this sampling. The values of $G^{\max}$, $g^{\max}$, $\beta_0^{\min}$ and $\beta_0^{\max}$ are provided in Table 4. After sampling global labels, we assign per-component genera such that they sum exactly to the global genus. This is achieved via a backtracking procedure (Algorithms 2–3).

Algorithm 1 Sampling (β_0, g)
Input: $g^{\max}$, $G^{\max}$, $\beta_0^{\min}$, $\beta_0^{\max}$, k

let $\mathfrak{B}_0 \leftarrow \{\underbrace{\beta_0^{min}, \ldots, \beta_0^{min}}_{\times k}, \underbrace{\beta_0^{min}+1, \ldots, \beta_0^{max}}_{\times k}\}$
initialize output list $P \leftarrow []$
for all $\beta_0 \in \mathfrak{B}_0$ **do**
  $g_{max} \leftarrow \min\left(G^{max},\ \beta_0 \cdot g^{max}\right)$
  accepted $\leftarrow$ false
  while not accepted **do**
    sample $s \sim \mathcal{U}[\![0, G^{max}]\!]$
    if $s \leq g_{\max}$ **then**
      accepted $\leftarrow$ true
    end if
  end while
  append (β_0, s) to P
end for
return P

6.1.2. Shape generation

Each component belongs to one of three categories: superquadrics, k-tori, or cones. We generate each family independently.

Superquadrics. We employ superellipsoids and supertoroids. Starting from a sphere or torus mesh generated with `Trimesh`, we apply the standard parametric deformation:

Algorithm 2 ENUMERATE-SOLUTIONS: Here a and b represent the number of components and the total genus respectively. Given an input configuration, previously determined (Algorithm 1), we enumerate all possible decompositions of the total genus into per-component genera. We further randomly pick one of them to actually create sample.

Input: a, b, g^{max}
Output: S

$S \leftarrow \emptyset$ ▷ Initialize solution set
if $b < 0$ **or** $b > g^{max} \cdot a$ **then**
  return S
end if
BACKTRACK$(a, b, 0, \emptyset, S)$ ▷ Start recursive enumeration
return S

Algorithm 3 BACKTRACK: Exploring all the possible decompositions of the total genus into per-component genera boils down to a tree search problem with backtracking once we have reached the maximum genus.

Input: r_{count}, r_{sum}, k, $\mathbf{x}$, S
Output: S

if $k > g_{\max}$ **then** ▷ Base case: all template types processed
  if $r_{\text{count}} = 0$ **and** $r_{\text{sum}} = 0$ **then**
    $S \leftarrow S \cup \{\mathbf{x}\}$
  end if
  return
end if ▷ Calculate upper bound for current template type
if $k = 0$ **then**
  $u_k \leftarrow r_{\text{count}}$
else
  $u_k \leftarrow \min(r_{\text{count}}, \lfloor r_{\text{sum}}/k \rfloor)$
end if ▷ Try all feasible counts for template type k
for $n_k = 0$ **to** u_k **do**
  $\mathbf{x}' \leftarrow \mathbf{x} \cup \{n_k\}$
  BACKTRACK$(r_{\text{count}} - n_k,\ r_{\text{sum}} - k \cdot n_k,\ k + 1,\ \mathbf{x}',\ S)$
end for

$$\text{Ellipsoid} \quad \begin{cases} x(u,v) &= s_x\, C_{\epsilon_1}(v)\, C_{\epsilon_2}(u) \\ y(u,v) &= s_y\, S_{\epsilon_1}(v)\, S_{\epsilon_2}(u) \\ z(u,v) &= s_z\, S_{\epsilon_1}(v) \end{cases} \tag{7}$$

$$\text{Toroid} \quad \begin{cases} x(u,v) &= s_x \left(R + C_{\epsilon_1}(v)\right) C_{\epsilon_2}(u) \\ y(u,v) &= s_y \left(R + S_{\epsilon_1}(v)\right) S_{\epsilon_2}(u) \\ z(u,v) &= s_z\, S_{\epsilon_1}(v) \end{cases} \tag{8}$$

where (s_x, s_y, s_z) are scale factors, (ϵ_1, ϵ_2) control shape sharpness, and $C_\epsilon(\cdot), S_\epsilon(\cdot)$ denote exponentiated trigonometric functions:

$$\begin{aligned} C_\epsilon(u) &= \text{sign}(\cos(u)) \,|\cos(u)|^\epsilon \\ S_\epsilon(u) &= \text{sign}(\sin(u)) \,|\sin(u)|^\epsilon \end{aligned} \tag{9}$$

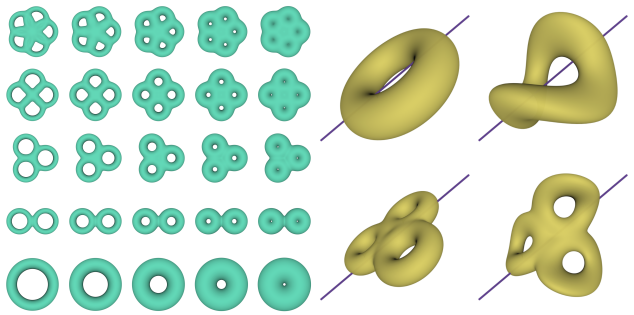

Figure 10. *(left)* Examples of k-tori for $k \in \{1, \ldots, 5\}$. *(right)* Twisting applied to 1- and 3-tori.

k-tori. Since no closed parametric form exists for a torus with k holes, we construct them via signed distance functions (SDFs). We generate k individual torus SDFs, combine them using the softmin,

$$\text{softmin}_k(s_1, s_2, \ldots, s_n) = -\frac{1}{k} \log \left(\sum_{i=1}^{n} e^{-k s_i} \right) \tag{10}$$

and extract the final mesh using marching cubes (Fig. 10).

Cones. Cone meshes are obtained directly using `Trimesh`.

6.1.3. Samples variety

To avoid geometric bias, we randomize all shape hyperparameters (e.g., scales, superquadric exponents, major/minor radii) within predefined ranges. We further apply random rigid motions and twisting deformations (Fig. 10, right) to each component before merging.

6.2. Baselines

Implementation. All baselines are trained from scratch on DONUT (Table 1) using their official implementations and hyperparameters.

Training. We train every model for 200 epochs with batch size 32 using Adam with initial learning rate 10^{-3}, reduced by a factor 0.5 every 20 epochs. We apply random rotations, translations, and scaling.

6.3. Implementation and training of FILTR

6.3.1. Input processing

All experiments use point clouds subsampled to 1024 points and normalized within the unit sphere. Persistence diagrams and all topological computations are performed using `Gudhi` [36].

6.3.2. Pretrained encoders

All encoders follow the same architecture and differ only by their pretraining method [10, 34, 56, 60]. Point clouds are partitioned into 64 patches of 32 points. Each patch is embedded into a 384-dimensional vector via a shared MLP, and patch centroids are mapped to positional encodings through another MLP. A 12-block transformer processes the resulting sequence. We use the pretrained checkpoints released by the respective authors.

6.3.3. Adapter

We map encoder outputs to the decoder space by applying layer normalization followed by a linear projection from 384 to 256 dimensions. Positional encodings are projected separately with a linear layer.

6.3.4. Transformer decoder

We adopt the DETR decoder architecture: 6 transformer blocks with self- and cross-attention, hidden dimension 256, and 8 attention heads.

6.3.5. Baselines for FILTR

For PointNet and DGCNN, we extract the per-point features prior to their global pooling stage and feed them to a 3-block transformer encoder with hidden dimension 256. For PointNet++ and RepSurf, whose architectures progressively downsample the point cloud through pooling, we instead use the features obtained after the second PointNet Set Abstraction layer (128 points). Using only the final globally pooled feature vector from the third abstraction layer led to unstable training. Positional embeddings are computed using an MLP: from each point of the input point cloud for PointNet and DGCNN, and from the pooled 128-point representation for PointNet++ and RepSurf. Table 5 reports the parameter counts for all FILTR variants.

6.3.6. Training

Models are trained on 23 579 DONUT point clouds using a single NVIDIA L40 GPU. We use batch size 64, train for 250 epochs with 5 warm-up epochs, and apply cosine learning-rate decay. We use $N = 250$ queries and optimize using AdamW with initial learning rate 10^{-4}. Loss weights are $\mu_{\text{recon}} = 1.0$, $\mu_{\text{exist}} = 0.1$, $\mu_{\text{diag}} = 0.1$; matching costs use $\lambda_{\text{reg}} = 1.0$ and $\lambda_{\text{exist}} = 0.1$. No data augmentation is applied. Pretrained encoders remain frozen; baseline models are trained end-to-end (Fig. 8).

Model	#params ($\times 10^6$)
FILTR + pretrained	5.7
FILTR + PointNet	8.1
FILTR + DGCNN	8.7
FILTR + PointNet++	7.9
FILTR + RepSurf	7.9

Table 5. **Number of trainable parameters for FILTR with different encoders.** Training an end-to-end pipeline adds around 2.5 million parameters compared to using a frozen pretrained encoder.

7. 3D vs. latent prediction pretraining

In this work, we focus on encoders pretrained with a 3D reconstruction objective. This choice is motivated by the geometric guarantees naturally provided by optimizing spatial reconstruction metrics.

7.1. Theoretical justification

Let X and $\hat{X} \in \mathbb{R}^{N\times 3}$ be the ground-truth and reconstructed point clouds. 3D-prediction encoders minimize the mean Chamfer distance (CD) between X and $\hat{X}$. To connect this objective to topological stability, we first relate CD to the Hausdorff distance (d_H), which measures the maximum spatial discrepancy between the two point sets.

Because the unreduced sum of minimum distances upper-bounds the maximum error, the Hausdorff distance is bounded by the mean Chamfer distance scaled by the number of points N:

$$d_H(X, \hat{X}) \leq N \cdot CD(X, \hat{X}) \tag{11}$$

Furthermore, the stability theorem for persistence diagrams establishes that the Bottleneck distance (d_B) between the persistence diagrams $D(X)$ and $D(\hat{X})$ is bounded by the Hausdorff distance:

$$d_B(D(X), D(\hat{X})) \leq d_H(X, \hat{X}) \tag{12}$$

Combining these inequalities yields:

$$d_B(D(X), D(\hat{X})) \leq N \cdot CD(X, \hat{X}) \tag{13}$$

Therefore, minimizing the Chamfer distance explicitly bounds the topological error. This mathematical guarantee ensures that learned features optimized for 3D reconstruction carry sufficient information to reconstruct persistence diagrams. In contrast, latent-prediction methods lack this geometric constraint. *Note:* The bounds derived above hold for the α-fitlration, but the general results still hold for the Vietoris-Rips filtration, albeit with a different constant factor.

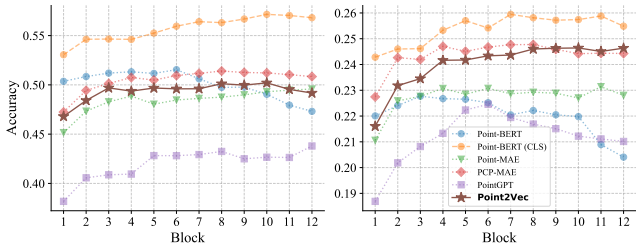

Figure 11. **Probing results on Point2Vec, compared to other encoders.** Point2Vec follows the same trend as other encoders.

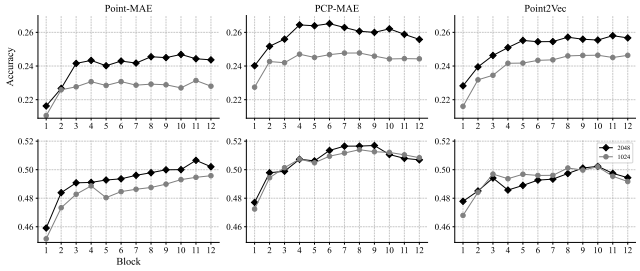

Figure 12. **Probing with different point cloud densities.** We report probing accuracies for Point-MAE, PCP-MAE, and Point2Vec on features computed from 1024- and 2048-point clouds. (top row) genus, (bottom row) connected components.

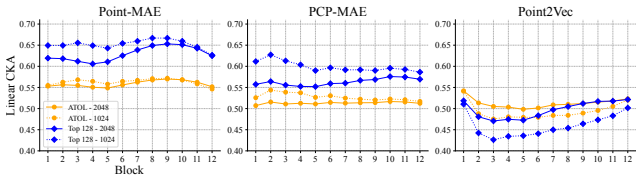

Figure 13. **CKA results with different point cloud densities.** We report alignment scores for Point-MAE, PCP-MAE, and Point2Vec on features computed from 1024- and 2048-point clouds.

7.2. Results on Point2Vec

Figure 11 highlights that probing results on Point2Vec features are comparable with Point-MAE and PCP-MAE. This indicates that encoders, regardless of their pretraining objectives (3D or latent prediction), capture a similar amount of global structural information. However, alignment scores in Figure 13 show that Point2Vec lags behind its 3D reconstruction-based counterparts. This suggests that latent-prediction encoders struggle to capture local topology, which strictly aligns with the theoretical guarantees discussed in Sec. 7.1.

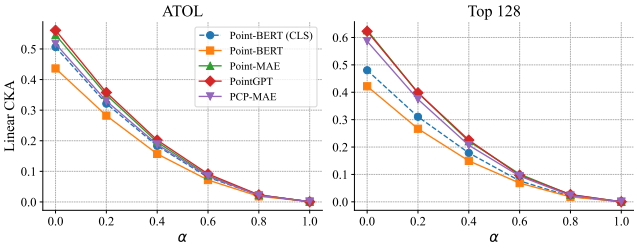

Figure 14. **CKA under controlled feature mismatch.** CKA similarity between the last transformer block of each encoder and ATOL/top-128 vectorizations on DONUT. A fraction α of features is randomly permuted, and results are averaged over 3 runs.

8. Experiments

8.1. Per-category probing results

Table 6 reports per-category probing accuracies along with baseline results. As expected, accuracy generally decreases for categories with higher topological complexity. Although Fig. 5 shows that probing performance tends to improve in deeper transformer blocks, the depth of the best-performing block (indicated in subscript) does not exhibit a consistent relationship with category difficulty. Finally, RepSurf [39] clearly outperforms all models trained from scratch, suggesting that explicitly encoding surface-based features provides a substantial advantage for capturing the underlying topology of point clouds.

8.2. Additional results on denser point-clouds

We discuss how probing and CKA results change when using encoder features computed from 2048-point clouds (instead of 1024). Intuitively, denser point clouds carry more information about the global structure of the shape, by "filling" space between points of 1024-point clouds. Therefore, fine topological structures are more salient. Figure 12 shows that genus prediction accuracy benefits from denser point-clouds, while connected components prediction remains similar. It is expected, since detecting connected components relies less on how densely sampled the shape is. This also reveals that, despite yielding rather poor probing scores, encoders implicitly carry some topological information about higher-order structures, that can be disambiguated with denser point clouds.

8.3. Relevance of CKA scores

The CKA similarities in Figure 6 allow comparison between encoders, but do not directly indicate whether the absolute CKA values represent meaningful alignment. Because CKA can be influenced by feature dimensionality and background correlations, we validate its interpretability through a controlled perturbation.

Let, $\{f_i\}_{i=1}^n$ denote the features extracted by the encoder and $\{v_i\}_{i=1}^n$ the corresponding vectorized persistence

Model	Genus											Connected Components					
	0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6
Pretrained-Frozen encoders																	
Point-BERT [56]	$53.3_{(9)}$	$27.2_{(7)}$	$27.5_{(3)}$	$19.0_{(3)}$	$17.7_{(10)}$	$23.3_{(3)}$	$12.8_{(5)}$	$12.0_{(7)}$	$16.2_{(5)}$	$20.7_{(1)}$	$20.8_{(5)}$	$84.1_{(3)}$	$61.3_{(3)}$	$43.7_{(4)}$	**$34.8_{(6)}$**	$31.8_{(10)}$	**$55.8_{(6)}$**
Point-MAE [34]	$56.2_{(10)}$	**$33.9_{(5)}$**	$28.3_{(7)}$	$19.3_{(6)}$	**$19.8_{(3)}$**	$23.5_{(2)}$	**$13.0_{(8)}$**	$13.2_{(12)}$	$15.8_{(1)}$	$22.0_{(10)}$	$21.3_{(7)}$	$84.2_{(4)}$	$60.0_{(12)}$	$40.2_{(5)}$	$33.0_{(8)}$	$30.7_{(11)}$	$53.9_{(9)}$
PointGPT [10]	$51.6_{(12)}$	$29.0_{(12)}$	$27.0_{(10)}$	$17.8_{(6)}$	$17.7_{(10)}$	$22.2_{(12)}$	$10.7_{(2)}$	$11.9_{(5)}$	$16.0_{(10)}$	$22.1_{(6)}$	$22.5_{(3)}$	$77.6_{(12)}$	$50.6_{(12)}$	$34.1_{(12)}$	$25.0_{(8)}$	$30.1_{(6)}$	$48.9_{(12)}$
PCP-MAE [60]	**$56.6_{(4)}$**	$33.4_{(5)}$	**$30.7_{(10)}$**	**$20.3_{(3)}$**	$19.6_{(7)}$	**$26.0_{(10)}$**	$12.9_{(2)}$	**$15.8_{(3)}$**	**$18.0_{(5)}$**	**$23.7_{(4)}$**	**$23.3_{(5)}$**	**$86.0_{(8)}$**	**$64.3_{(7)}$**	**$44.1_{(8)}$**	$34.4_{(12)}$	**$33.6_{(5)}$**	$53.6_{(7)}$
Baseline models trained from scratch																	
PointNet [37]	55.7	30.8	13.4	8.00	8.92	23.1	4.18	1.40	5.88	**46.9**	13.3	89.1	71.5	41.4	31.7	26.9	52.1
PointNet++ [38]	89.8	63.5	64.7	55.6	52.0	50.4	22.2	32.1	25.3	38.0	53.8	99.6	94.5	81.7	61.4	47.5	65.9
DGCNN [48]	80.0	63.5	47.7	46.3	33.2	37.4	13.8	21.4	10.4	28.1	52.4	99.6	93.9	83.3	71.3	**59.2**	72.7
RepSurf [39]	**93.0**	**70.3**	**77.5**	**64.5**	**64.0**	**54.7**	**30.5**	**29.8**	**26.7**	32.6	**63.1**	**100**	**97.4**	**89.3**	**74.4**	54.8	**82.8**

Table 6. **Performance per category on DONUT.** We report classification accuracies (%) for genus and number of connected components prediction, for pretrained encoders and baseline models trained from scratch on DONUT. For pretrained encoders, we indicate in subscript the transformer block that achieved the best accuracy.

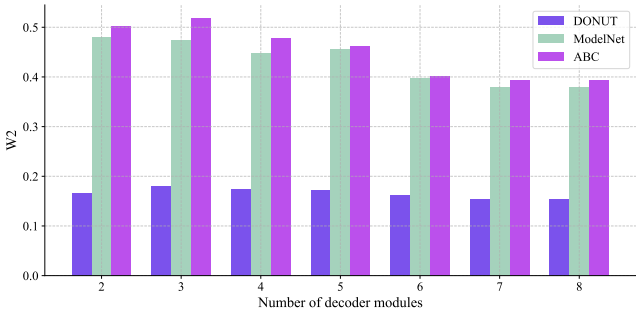

Figure 15. **Effect of decoder depth.** We train FILTR on DONUT with varying decoder depth using a Point-MAE backbone. We report 2-Wasserstein distances on DONUT (test), ModelNet, and ABC.

descriptors, with a one-to-one correspondence between indices. For a given proportion $\alpha \in [0, 1]$, we introduce a permutation $\sigma^{(\alpha)}$ that randomly permutes a fraction α of the indices and therefore creates mismatches. We then compute $\mathrm{CKA}(f_{\sigma^{(\alpha)}(i)}, v_i)$ as a function of α.

Figure 14 shows the resulting degradation for ATOL and top-128 vectorizations. The rapid decline in similarity confirms that high CKA values cannot be explained by dimensionality alone and instead reflect genuine structural alignment between learned features and persistence information.

8.4. Additional results on FILTR

Model	FLOPS ($\times 10^9$)	Allocated Memory (GB)	Training (hours)	Inference (ms)
FILTR + pretrained	4.09	2.36	9	25.5± 0.1
FILTR + PointNet	14.64	11.43	19	140± 0.1
FILTR + DGCNN	19.10	15.45	22	230± 0.2

Table 7. **Computational Cost.** FLOPS are estimated on a single input sample. Training setup is similar to the one used in the main paper. Inference time is estimated for a batch size of 64.

Decoder depth. To evaluate the role of decoder depth, we train FILTR with different numbers of transformer decoder blocks while keeping all other hyperparameters fixed. As shown in Fig. 15, performance on ModelNet and ABC improves up to six decoder blocks, after which additional depth yields diminishing returns.

Loss	$W_{2\,(\times 10^{-2})}$	$d_{B\,(\times 10^{-3})}$	PIE
$\mathcal{L}_{\text{recon}}$	23.63	10.47	4.079
$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{exist}}$ w/ p_e	16.42	9.866	1.193
$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{exist}}$ w/o p_e	43.06	10.75	–
$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{exist}} + \mathcal{L}_{\text{diag}}$ w/ p_e	17.27	9.917	1.107
$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{exist}} + \mathcal{L}_{\text{diag}}$ w/o p_e	17.24	9.918	–

Table 8. **Ablation study of losses.** We use a Point-MAE encoder that achieves competitive results on DONUT (Tab. 3). Similar results are observed with other encoders (see Tab. 9). We report results for both thresholded (w/ p_e) and non-thresholded (w/o p_e) existence probability p_e when using the existence loss $\mathcal{L}_{\text{exist}}$.

Loss	Point-BERT	PCP-MAE	PointGPT
$\mathcal{L}_{\text{recon}}$	31.67	26.62	53.73
$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{exist}}$ w/o p_e	113.6	67.64	291.72
$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{exist}}$ w/ p_e	17.33	**17.02**	18.20
$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{exist}} + \mathcal{L}_{\text{diag}}$ w/o p_e	16.23	17.63	17.95
$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{exist}} + \mathcal{L}_{\text{diag}}$ w/ p_e	**16.18**	17.18	**17.86**

Table 9. **Ablation study of losses on additional encoders.** We report $W_{2\,(\times 10^{-2})}$ for FILTR trained with different pretrained encoders.We report results for both thresholded (w/ p_e) and non-thresholded (w/o p_e) existence probability p_e when using the existence loss $\mathcal{L}_{\text{exist}}$.

Ablations. Table 8 shows that $\mathcal{L}_{\text{exist}}$ substantially improves reconstruction, compared to $\mathcal{L}_{\text{exist}}$ alone. As expected, without $\mathcal{L}_{\text{diag}}$, it is essential to threshold non-existing persistence pairs to get good performance. Finally, while introducing $\mathcal{L}_{\text{diag}}$ slightly reduces performance for W_2, they remain identical with and without thresholding. We also note that overall, the Bottleneck distance d_B seems to be hardly impacted by these variants. Furthermore, Table 9 extends the loss ablation experiments by reporting 2-Wasserstein distances for all remaining encoders.

Dataset	$W_{2\,(\times 10^{-2})}$	$d_{B\,(\times 10^{-3})}$	PIE
DONUT	36.17	10.26	13.26
ModelNet	56.37	13.07	14.28
ABC	73.77	32.15	5.92

Table 10. **Reconstruction results with RepSurf.**

8.4.1. Discussion on PointNet++

To adapt PointNet++ and RepSurf to our setting, we use the 128 per-region features produced after the second Set Abstraction layer, before the final pooling stage, as input to the FILTR decoder. These intermediate features preserve local geometric information while being stable enough to train effectively, in contrast to using the final globally pooled representation, which led to unstable training.Table 10 shows the performance of FILTR with RepSurf as feature extractor.

8.4.2. Performance of DGCNN baseline

Extractor	$W_{2\,(\times 10^{-2})}$	$d_{B\,(\times 10^{-3})}$	PIE
Point-MAE$_C$	29.74	10.54	3.874
DGCNN (E2E)	$113.1_{(+83.36)}$	$36.67_{(+26.13)}$	$8.301_{(+4.427)}$

Table 11. **Results on DONUT under low data regime (2K shapes).** The two first rows compare frozen and E2E training

As pointed in Section 4.5, the end-to-end (E2E) baseline with DGCCN feature extractor tends to outperform pretrained feature extractors (Tab. 3) on ModelNet and ABC. We hypothesize that this stems from two reasons: (1) E2E models naturally excel in *high-data regimes* by overfitting to task-specific distributions. However, FILTR with frozen encoders demonstrate widely superior performance in *low-data regimes*. As shown in Table 11, FILTR significantly outperforms E2E DGCNN. (2) DGCNN's architecture is explicitly tailored to capture local topology of point clouds– as claimed by the authors–making it biased towards this task.

8.4.3. Results with Vietoris-Rips Filtration

While our primary experiments utilize the $alpha$-filtration, FILTR is fundamentally agnostic to the specific choice of filtration. Because the architecture treats persistence diagrams strictly as unordered sets, it only requires the resulting persistence pairs as target inputs during training. Consequently, the model is fully capable of learning and fitting the specific data distribution of the target diagrams regardless of the underlying mathematical method used to compute them. To empirically demonstrate this flexibility, we train and evaluate (Tab. 12) FILTR on a subset of 2K samples from the DONUT dataset on Vietoris-Rips filtration.

8.4.4. Qualitative results

Reconstruction. Figure 16 shows that FILTR captures the overall structure of persistence diagrams across datasets.

Extractor	$W_{2\,(\times 10^{-2})}$	$d_{B\,(\times 10^{-3})}$	PIE
Point-MAE$_C$ (Rips)	92.55	41.23	8.562

Table 12. **Results on DONUT under low data regime (2K shapes) for Vietoris-Rips filtration.** FILTR is trained on Point-MAE (combined features) to predict VR persistence diagrams. While *no quantile thresholding* is applied to the predicted diagrams, the model still achieves competitive performance with the E2E baseline (Tab. 11).

The predicted distributions and magnitudes of persistence pairs generally align with the ground truth. However, as discussed in Section 3.2, the most persistent pairs, which correspond to the dominant topological features of a shape, remain difficult to predict accurately. Estimating these pairs requires the encoder to capture global geometric structure, a capability that pretrained models struggle with, as indicated in Table 1.

Figure 17 illustrates typical failure cases. The most common error is a shift between the predicted and ground-truth locations of persistence pairs. This effect appears on both ModelNet and ABC, but is more pronounced on ABC, where mismatches may span several orders of magnitude. This behavior is consistent with the distribution shift between datasets: pretrained encoders are primarily exposed to ShapeNet-like geometry during pretraining, while ABC shapes exhibit topological configurations that are not well represented in ShapeNet.

Effect of the diagonal loss. Figure 18 illustrates the impact of including the diagonal loss term $\mathcal{L}_{\text{diag}}$ in FILTR's training objective. Without this term, the model tends to produce persistence diagrams with a higher density of low-persistence points near the diagonal. They require using the existence probability to filter noisy points and retrieve accurate diagrams. With the diagonal loss, diagrams produced without using the existence probability remain close to the ground-truth one (Tab. 9).

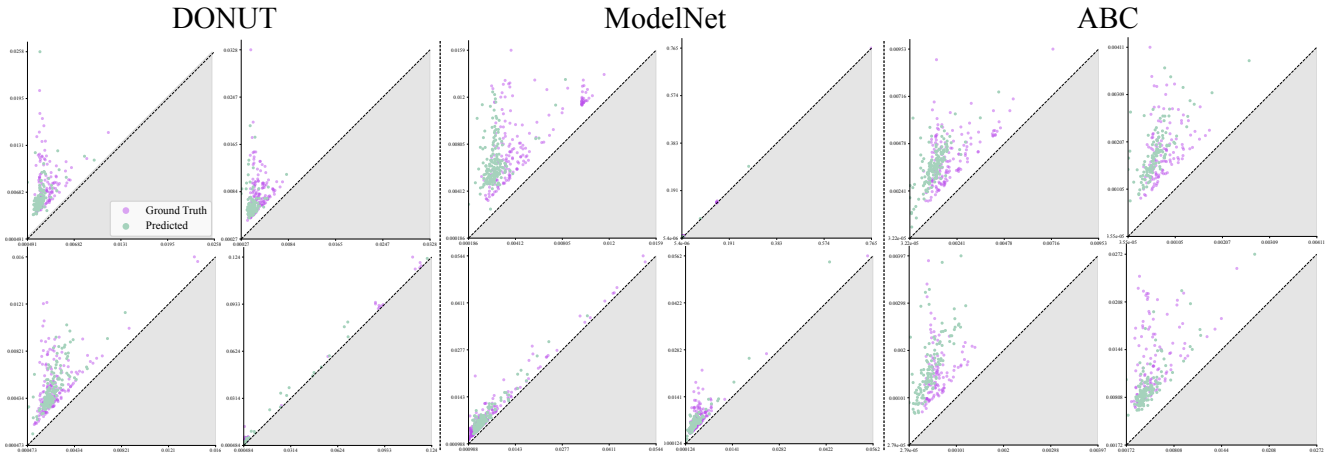

Figure 16. **Predicted persistence diagrams.** Predicted vs. ground-truth persistence diagrams from FILTR (Point-MAE backbone) on DONUT, ModelNet, and ABC samples.

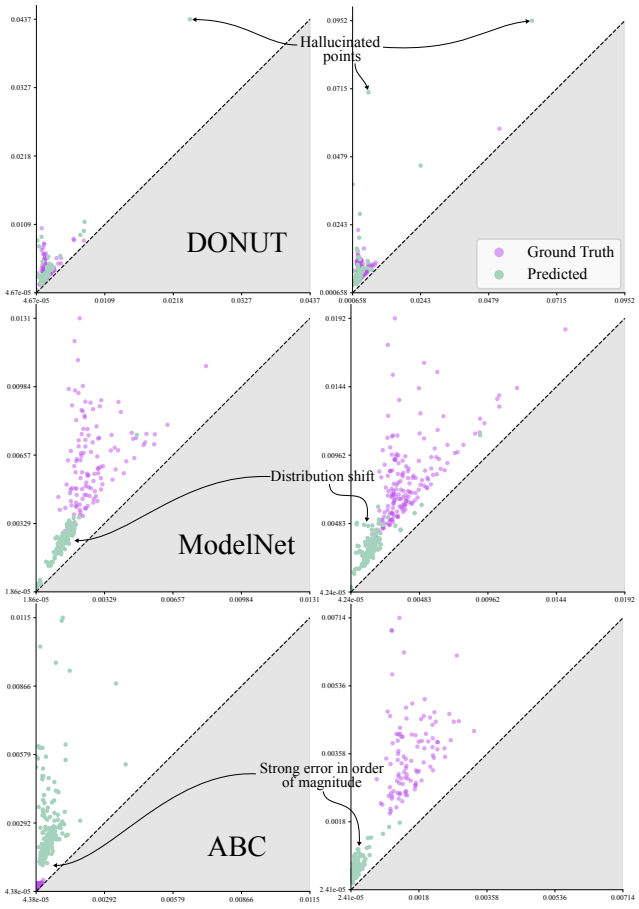

Figure 17. **Failure cases.** Predicted vs. ground-truth persistence diagrams from FILTR (Point-MAE backbone) on DONUT, ModelNet, and ABC samples.

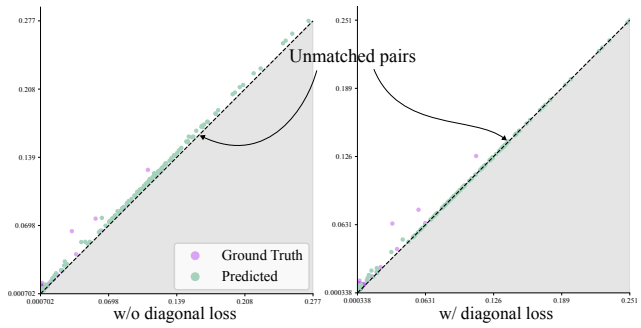

Figure 18. **Effect of $\mathcal{L}_{\text{diag}}$.** *(left)* Unmatched pairs a close to the diagonal but still contributing to the 2-Wasserstein distance. *(right)* With the diagonal loss, unmatched pairs are exactly on the diagonal, contributing zero to the distance.