Data-Driven Estimation of Topological Features in 3D shapes

Author:
Louis Martinez

Télécom Paris

Supervisor:
Maks Ovsjanikov

LIX, Ecole Polytechnique

October 2025

Contents

Abstract

Large transformer-based foundation models have shown remarkable ability to extract expressive representations in a purely self-supervised, data-driven manner, and recent efforts have extended them to 3D shape understanding. In parallel, Topological Data Analysis (TDA), through the lens of persistent homology, offers mathematically grounded tools to characterize the structural information carried by point clouds. However, persistent homology scales poorly with dimension and sample size, limiting its applicability to large datasets. This thesis makes two main contributions. First, we investigate to what extent pretrained 3D transformer models capture structural, non-semantic properties of shapes, using TDA as a principled framework to quantify their topological awareness. Second, building on these insights, we introduce a data-driven proxy for persistent homology: a transformer-based approach that leverages learned representations to predict topological features efficiently. Our results highlight both the limitations of current 3D transformers and the potential of bridging foundation models with TDA to advance topology-aware 3D representation learning.

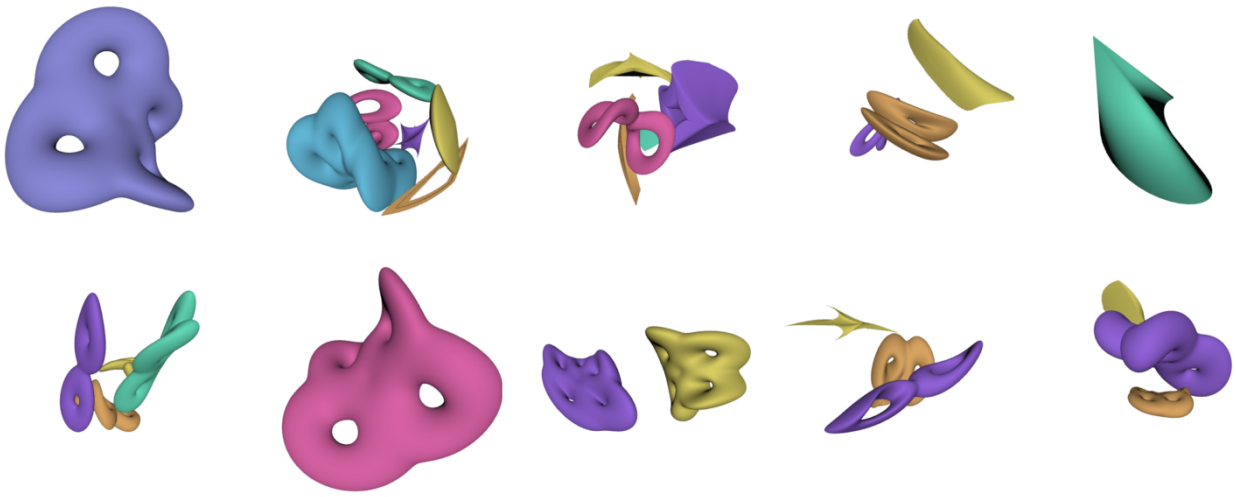


Figure 1: **Random samples from the dataset.** Despite the fact that each sample is generated from a rather small family of shapes, both the topology preserving placement and augmentations allow for a rich variety of shapes, while ensuring topological consistency.

1 Introduction

Foundation models for 3D data have recently emerged a promising direction for scaling 3D shapes analysis. Inspired by the success of large language models and images encoders, especially the expressivity of the learned representations, this paradigm has been adapted to 3D shapes, leading to significant improvements in various tasks such as shape classification, segmentation, and generation. These models are pretrained on vast amounts of unlabeled 3D data to learn rich representations that can be further fine-tuned for downstream tasks, enabling better generalization. Although these models can capture semantic and geometric information from 3D shapes quite well, it remains unclear to what extent they also encode non-semantic, structural information, such as the topology. More generally, pretraining procedures for 3D shapes encoders have been directly adapted from the image and text domains, without considering the specific characteristics of 3D data. As a consequence, sate-of-the-art methods have a fairly limited range of validity. For instance, models trained on object centric data can't generalize to larger scenes, while models aimed to process large scenes struggle in capturing fine-grained details. This raises the question of whether these models are truly capable of capturing the unique properties of 3D shapes at different scales, or if they simply learn to mimic the patterns found in the training data. The actual quality of the learned representations remains an open question. In this report, we aim to shed light on these issues by investigating the extent to which 3D shapes encoders capture structural information, and how this impacts their performance across different tasks and scales. We also explore the limitations of current pretraining procedures and propose potential improvements to enhance the robustness and generalization capabilities of 3D shapes encoders.

The contributions of this report are 3-folds:

- We introduce DONUT, a scalable dataset of 3D shapes with full control over the topology of the generated 3D shapes
- We carry out comprehensive experiments to understand and quantify to what extent current state-of-the-art 3D shape encoders capture structural, non-semantic information from 3D shapes.
- Guided by the results obtained from our experiments, we propose architectural improvements and pretraining strategies to encourage the presence of such properties.

2 DONUT: Dataset Of maNifold strUcTures

Understanding how models capture topological properties of 3D shapes is a key step toward disentangling geometric representation learning from semantic categorization. Probing such capabilities requires datasets

with explicit topological labels and fine-grained control over their distribution. However, existing 3D datasets, such as Objaverse [9, 10] or ShapeNet [6], focus primarily on semantic categories or low-level geometry, and do not provide systematic coverage of topological variation.

We introduce DONUT, a scalable dataset of synthetic 3D shapes annotated with accurate and balanced topological labels. Unlike prior datasets, DONUT is designed specifically to isolate intrinsic topological invariants, such as the number of connected components and the genus, enabling controlled evaluation of how models represent topology. This resource provides a foundation for probing, benchmarking, and training models on tasks where topological understanding is essential.

2.1 Existing datasets

Existing 3D datasets provide only limited support for probing whether models capture topological structure. Broadly, they fall into three categories: combinatorial-only benchmarks, geometric datasets with noisy topology, and synthetic datasets with limited diversity.

Combinatorial benchmarks. The MANTRA dataset (Manifold Triangulation Assemblage) [2] provides combinatorial triangulations of 2D and 3D manifolds, represented as abstract simplicial complexes without embedding in $\mathbb{R}^3$. MANTRA is a valuable testbed for assessing whether graph- or simplicial complex models capture higher-order structures such as Betti numbers $(\beta_0, \beta_1, \beta_2)$. However, since it lacks any geometric realization, MANTRA is not suitable for evaluating models built on geometric 3D representations such as meshes, point clouds, or implicit fields.

Dataset	#Samples	Meshes	Manifold	Balanced annot.
DONUT	30,000	✓	✓	✓
MANTRA [2]	43,100	–	✓	–
ABC [23]	1,000,000+	✓	–	–
Thingi10K [41]	10,000	✓	–	–
EuLearn [12]	3,300	✓	✓	✓

Table 1: **Overview of existing datasets and their capabilities.** We summarize here the main characteristics of existing datasets with topological annotations. Besides EuLearn, all existing datasets with topological annotations come with downsides, discussed in Section 2.1. However, since EuLearn seems to be the most promising dataset, we carried out an extensive analysis to (1) highlight limitations that make it unreliable for further experiments and (2) motivate the use of DONUT (see Appendix A.1). *Note:* The number of samples for MANTRA only takes into account 2-manifolds.

Geometric datasets with noisy topology. Large mesh datasets such as Thingi10K [41], Objaverse [9] and ABC [23] contain CAD models and artistic objects, and include coarse topological annotations (number of components, genus). However, these annotations are unreliable for several reasons:

1. *Discrepancy between raw and perceptual components.* Artistic and CAD models are typically constructed from many sub-meshes, so the annotated component count often diverges from the semantically meaningful object count.

2. *Severe class imbalance.* While a wide range of genus values is theoretically possible, the overwhelming majority of meshes in both datasets have genus 0–2, making them unsuitable for balanced probing tasks.

3. *Structural artifacts.* Many meshes are non-manifold or self-intersecting, rendering quantities such as the Euler characteristic ill-defined:

$$\chi = V - E + F = 2 - 2g - b + c \tag{1}$$

where V, E, F are the number of vertices, edges, and faces, g is genus, b the number of boundary components, and c the number of connected components. Attempts to repair such meshes (e.g., Manifold [18], ManifoldPlus [19], DOGN [33], CLAY [38]) face tradeoffs between oversmoothing geometry, introducing artifacts, or incurring prohibitive computational cost. As a result, these datasets cannot provide reliable large-scale topological ground truth.

Synthetic datasets with limited diversity. The EuLearn dataset [12] addresses class imbalance by generating surfaces of varying genus from Fourier curves. While balanced across genera, EuLearn suffers from two critical issues: (i) samples within each genus class lack diversity, making them nearly indistinguishable, and (ii) strong correlations emerge between genus and canonical orientation, enabling trivial classification via nearest-neighbor with Chamfer distance. Consequently, EuLearn fails to disentangle topological structure from geometric shortcuts, limiting its usefulness for robust probing.

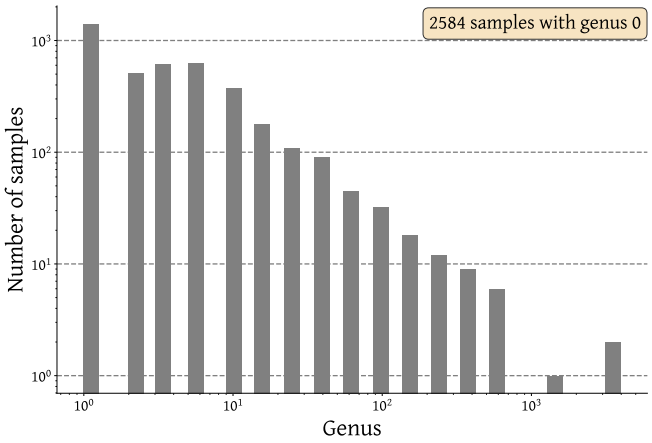


Figure 2: **Genus distribution of Thingi10K.** Both axes are plotted in log scale. The genus was estimated from the Euler characteristic, provided as metadata with the dataset; however, 2,651 samples do not fulfill the requirements to directly compute the genus from the Euler characteristic (see 1). They are therefore not taken into account here. The histogram shows that a large fraction of the dataset (over 3,000 samples out of 7,344) has genera of 0 or 1, indicating that higher-genus components are significantly underrepresented, which may limit accurate classification and probing analyses for those cases.

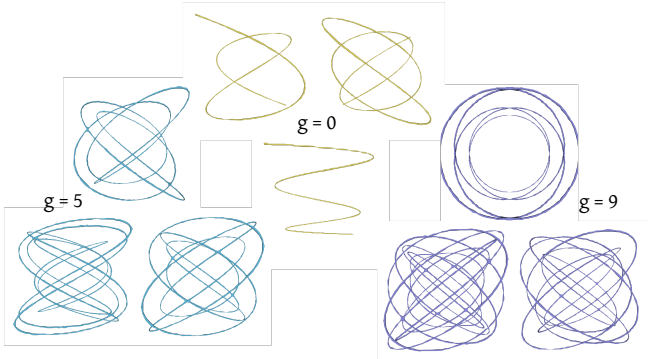


Figure 3: **Samples from the EuLearn dataset across different genera.** As the genus increases, shapes become geometrically more complex. This trend highlights a confounding factor in the dataset: geometric complexity grows together with genus. As a result, classification performance may be driven not only by topological information but also by correlated geometric cues.

2.2 Method

Since DONUT is made of synthetic shapes, two critical challenges must be addressed to ensure that the results obtained from this dataset are relevant. 1) Foundation models are pretrained on real shapes. Therefore, learned features reflect the distribution of real world geometry and semantics. Synthetic data, however, introduce patterns and biases that don't exist in real data, making them fall out-of-distribution. 2) Another common pitfall of synthetic datasets (cf. EuLearn) is the lack of variability, making evluation tasks such as probing, trivial. In practice, the first challenge is overcome by the fact that foundation models are pre-trained on datasets (e.g ShapeNet, Objaverse, etc.) that are large enough for synthetic shapes not to fall out-of-distribution. This is experimentaly validated in Figure (+ figure with umaps). To adress the second challenge, we developed a set of rules and augmentation techniques to ensure that the generated shapes are both diverse and topologically accurate.

Below we derive these rules used to generate DONUT, and more importantly, how the latter allow to maintain this variability across samples while scaling their number up to 10^5 shapes.

2.2.1 Labels Distribution

A central design choice in DONUT is to ensure balanced topological supervision across the dataset. To this end, both the genus and the number of connected components are sampled such that their marginal distributions are approximately uniform. This prevents any single topological class from dominating and guarantees that models trained on the dataset are exposed to the full spectrum of topological structures. We bring this guarantee over marginal distributions by first sampling the labels before actually generating any mesh. **??** formally describes how the labels are sampled and further details the properties of the dataset we used for all our experiments

Each sample is further composed of multiple meshes selected from a predefined family of template shapes. This choice introduces large geometric variability while maintaining strict control over label balance. As a result, the dataset couples statistical uniformity in labels with broad diversity in shape realizations. Nevertheless, by limiting ourselves to certain families of geometric shapes, we avoided falling into the pitfall of geometric complexity mentioned earlier.

2.2.2 Sample-Level Properties

At the level of individual samples, the generation process begins by selecting the global genus and number of connected components. These values are then distributed across the meshes composing the sample. For example, a sample may include one mesh of genus one, several genus-zero meshes, or even higher-genus meshes, depending on the chosen configuration.

Each mesh is then instantiated from a specific parametric family of shapes:

- *Genus 0:* Superellipsoids and cones
- *Genus 1:* Supertoroids and cones
- *Genus* ≥ 2*:* K-tori

Superquadrics. Superellipsoids (resp. toroids) are part of a wider family of parametric shapes called superquadrics. They were introduced by Barr et al. [3] in 1981 and are widely used in computer graphics for their ability to represent a large range of shapes with a small number of compact parameters. Superquadrics have recently regained attention in 3D scene understanding, notably in SuperDec [11], because of their expressiveness and their ability to approximate complex structures by composition. Their implicit equation is given by:

$$\left(\left| \frac{x}{s_x} \right|^{\frac{2}{\epsilon_2}} + \left| \frac{y}{s_y} \right|^{\frac{2}{\epsilon_2}} \right)^{\frac{\epsilon_2}{\epsilon_1}} + \left| \frac{z}{s_z} \right|^{\frac{2}{\epsilon_1}} = 1 \tag{2}$$

where $(s_x, s_y, s_z) > 0$ are scale factors along the x, y, z axes respectively, and $(\epsilon_1, \epsilon_2) > 0$ are shape exponents that modulate the surface's roundness (resp. sharpness). Sampling these parameters within predefined ranges allows to efficiently sample a variety of shapes while maintaining control over their topological properties.

2.2.3 Mesh Generation

One key challenge is to be able to generate a large amount of samples (up to 10^5) in a reasonable amount of time. This order of magnitude is motivated the results of Point-MAE-Zero [7]. Accurate representations were obtained with a pretraining set containing around 150K samples. While we further use DONUT essentially for evaluation/probing tasks, we want to preserve the possibility of scaling to pretraining regimes where data requirements are significantly higher. We therefore restrict as much as possible the use of computation intensive operations. k-tori aside, every mesh is generated either with its parametric expression when available, or with simple rules.

Cones. ...

Superquadrics. Meshes corresponding to super ellipsoids and super toroids are generated directly from their parametric forms. The procedure consists of two steps: 1) generate a base template mesh (a sphere for ellipsoids, a torus for toroids) where vertices are expressed in spherical (resp. toroidal) coordinates, and 2) deform it according to the superquadric equations. This construction is lightweight, parallelizable, and supports fast large-scale dataset generation. The parametric equations we use in practice are as follows:

$$\text{Ellipsoid} \quad \begin{cases} x(u,v) &= s_x\, C_{\epsilon_1}(v)\, C_{\epsilon_2}(u) \\ y(u,v) &= s_y\, S_{\epsilon_1}(v)\, S_{\epsilon_2}(u) \\ z(u,v) &= s_z\, S_{\epsilon_1}(v) \end{cases} \qquad \text{Toroid} \quad \begin{cases} x(u,v) &= s_x \left(R + C_{\epsilon_1}(v)\right) C_{\epsilon_2}(u) \\ y(u,v) &= s_y \left(R + S_{\epsilon_1}(v)\right) S_{\epsilon_2}(u) \\ z(u,v) &= s_z\, S_{\epsilon_1}(v) \end{cases} \tag{3}$$

where $u, v \in [-\pi, \pi]$ are the vertex coordinates of the template mesh is spherical (resp. toroidal) coordinates and:

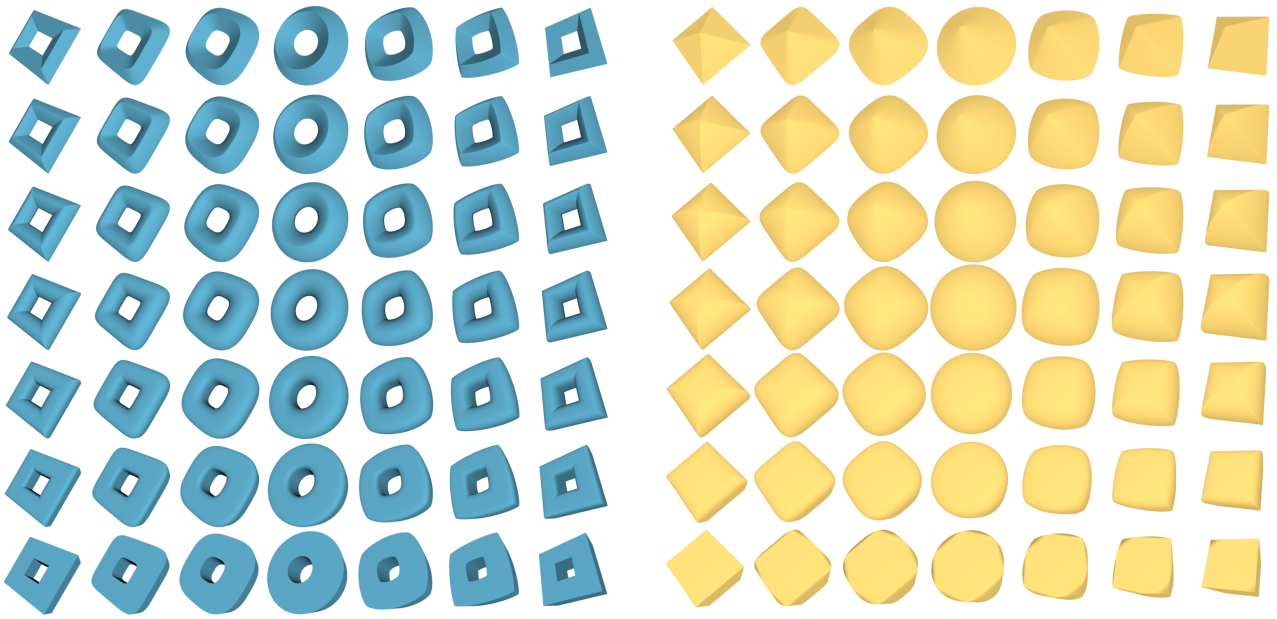
$$\begin{aligned} C_\epsilon(u) &= \text{sign}(\cos(u)) \,|\cos(u)|^\epsilon \\ S_\epsilon(u) &= \text{sign}(\sin(u)) \,|\sin(u)|^\epsilon \end{aligned} \tag{4}$$

K-tori. In the case of higher-genus meshes, there are no closed-form parametric equations. Instead, we leverage the implicit formulation of tori. A k-torus is generated by (1) evenly distributing tori on a circle, (2) blending their signed distance functions (SDF) through a smooth union operator, and finally (3) extracting the mesh via marching cubes with a grid-size. However, this process is more computationally intensive as it requires applying marching cubes the discretized SDF, whose complexity scales in $O(N^3)$ where N is the grid size. And since we seek to have high quality meshes to preserve fine-grained details (here holes), we use a higher grid resolution.

To smoothly blend the SDF of each torus generated independently, we use the *smooth minimum* operator:

$$\text{softmin}_k(s_1, s_2, \ldots, s_n) = -\frac{1}{k} \log \left(\sum_{i=1}^{n} e^{-k s_i} \right) \tag{5}$$

Where $(s_1, s_2, \ldots, s_n) \in \mathbb{R}^{(N^3)}$ are the SDF of individual tori and k is a hyperparameter controlling the smoothness of the blending. This operator is applied voxel wise. In other words, each voxel of the blended SDF is assigned with the smooth minimum value of the individual SDF.



(a) Supertoroids. (b) Superellipsoids.

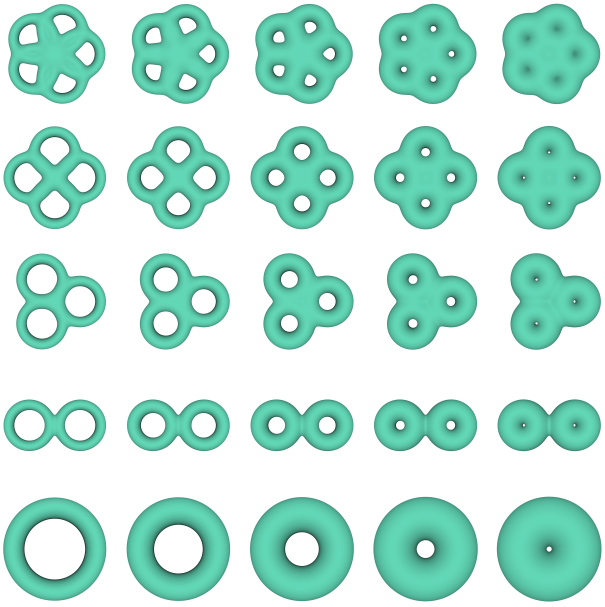
Figure 4: Overview of different shapes obtained for fixed values of $a_i, i \in \{1, 2, 3\}$ and increasing values of ϵ_1 (left to right) and ϵ_2 (bottom to top). (a) Different supertoroids. As mentioned in TO ADD, using these shapes for k-tori ($k \geq 2$) is challenging because they may not preserve the genus, for instance if some parts are too thin or sharp. (b) Different superellipsoids.

2.2.4 Topological Consistency and Diversity

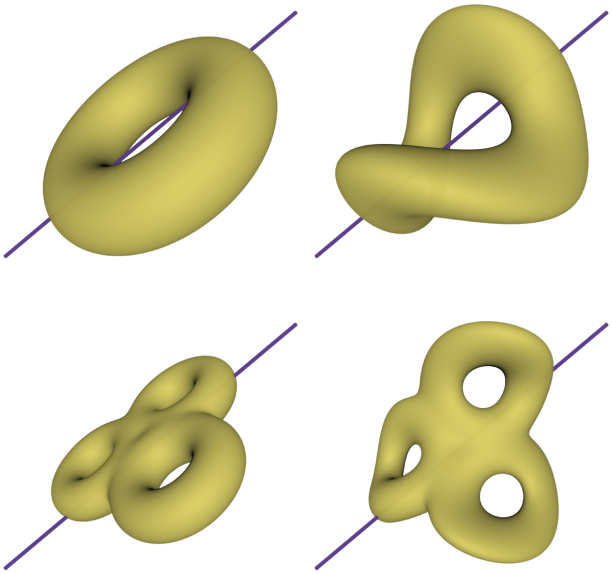
We still have two challenges addressing. First, when placing components within a sample, we must ensure they don't overlap. At the same time, they need to be close enough to make the samples both topologically consistent and challenging enough for models to correctly identify the number of connected components. In early experiments, we observed that if components are too far apart, the task can be trivially solved using a simple KNN classifier with Chamfer distance. Second, even with careful placement, models tend to overfit, sometimes even when using simple linear heads on top of learned features, showing that this alone is insufficient to evaluate their true topological understanding. To increase variability while preserving topological labels, we apply transformations both at the sample level and the individual component level.

Consistency. To address the first challenge, we developed a two-stage placement procedure. Suppose there are already some components correctly placed within a sample. To add a new component, we first place it randomly in the sample and then check for intersections with the existing components. Experimentally, we observed that a randomly placed mesh overlaps with at most two other components. If the new component overlaps with only one existing component, we identify the point on the new mesh that is most deeply inside the other component and push the new component along the opposite direction of the surface normal at that point. For intricate shapes, a single adjustment may create new overlaps, so we repeat this step five times. If overlaps remain after five attempts, the component is discarded and a new one is generated and randomly placed. This iterative procedure improves efficiency: in practice, one or two adjustments are sufficient to resolve overlaps, making it faster than discarding and regenerating components immediately.

Variability. To further increase the diversity of DONUT while preserving topological labels, we apply transformations both at the sample level and at the component level. Commonly used rigid transformations, such as rotations and scaling, are applied to both components and full samples. At the component level, we do not apply translations to avoid creating overlaps that would break the overall topology of the sample. To introduce additional geometric variability, we also apply twisting deformations (Figure 5b). Without loss of generality, we assume that a component (or a full sample) lies within the unit sphere, as global scaling can be applied afterward. First, we uniformly sample a direction on the unit sphere, which defines an axis for the



(a) **Degrees of freedom allowed on k-tori.** Prior to the sequence of transformations applied to each individual component (e.g. rotation, twisting) we allow some variability during the generation. The x-axis represents the ratio *major radius / minor radius*. The y-axis shows how the template shape is modified as we increase the value of k (i.e. the genus). *Note:* In the actual dataset, 1-tori are generated with the parametric representation of supertoroids, since a regular torus is a particular case of supertoroid. However, for $k \geq 2$, using a composition of 1-tori is more reliable. Some parameter sets can lead to undesirable holes in the final shape.



(b) **Effect of twisting.** Original template shapes (left) are twisted along the purple axis (right). Twisting deformations introduce non-rigid variability while preserving the genus and connectivity of each component. Here the scalar function defined along the axis is affine between $-\pi/6$ and $\pi/3$. We apply this augmentation at the component and sample level.

twist. Next, we define a smooth scalar function along this axis that determines the rotation angle. Finally, each vertex is rotated around the axis by the angle given by the scalar function evaluated at the point where the vertex projects along the axis. This procedure introduces non-rigid variability while preserving the genus and connectivity of each component and of the overall sample.

2.3 Analysis

This section provides a detailed characterization of DONUT. We first present dataset statistics, label distributions, and its relation to pretrained models. We then discuss baseline performance and introduce two complementary protocols designed to verify that models trained on DONUT actually capture topological features.

2.3.1 General Properties

Dataset statistics. DONUT contains 29,517 samples divided into training, validation, and test splits of 80%, 10%, and 10%. Each sample has between 1 and 6 connected components, with the total genus ranging from 0 to 10. A single component has genus at most 5 (corresponding to a 5-torus (Figure 5a)). The dataset size was chosen to balance feasibility and reliability: smaller datasets led to overfitting while much larger datasets made experiments costly. Figure 7a reports the marginal distributions of genera and connected components.

Distributions and scalability. Figure 6 shows that synthetic samples from DONUT occupy the same representation space as training data used by common 3D encoders. This indicates that the dataset lies within the learned distribution of existing models rather than being out-of-distribution. In addition, Figure 7b reports the time required to generate different dataset sizes, showing that large-scale variants remain practical.

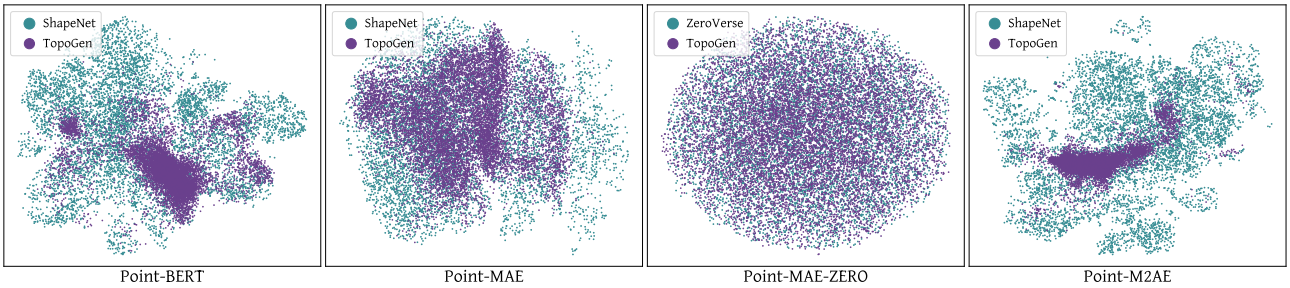


Figure 6: **UMAP embeddings of features learned by 3D encoders.** DONUT samples project into the same space as the original training data, indicating that they are not out-of-distribution.

2.3.2 Baselines and Transferability

We trained several classical point-cloud models (PointNet, PointNet++, DGCNN) and recent persistence-based models (PersFormer, xPerT, PersLay) on DONUT. Results are reported in Table 2. Beyond in-distribution accuracy, a key question is whether these models actually capture topology or only rely on geometric characteristics.

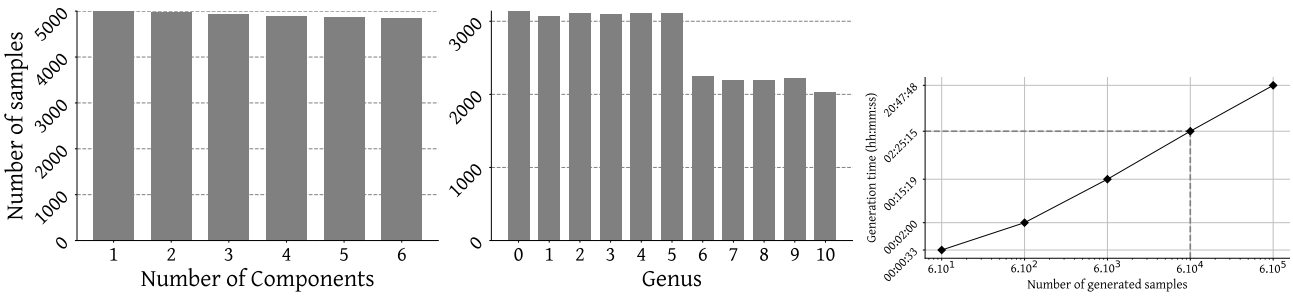
Transferability provides a natural probe. To test this, we evaluated models on a curated subset of ABC. This set contains shapes whose geometry differs significantly from the training distribution, but whose topology is well characterized and within the label space. If models have learned topology, they should retain reasonable performance on ABC despite geometric differences. Table 2 shows that DGCNN and specialized topological models maintain strong transfer, supporting the idea that DONUT encourages models to focus on topology. We note that weak transfer can also reflect limitations of current architectures rather than flaws in the dataset itself.

	Genus			**Connected Components**		
Model	MSE↓	Acc.↑	F1↑	MSE↓	Acc.↑	F1↑
			Point-based models			
PointNet [28]	–	–	–	–	–	–
PointNet++ [27]	–	–	–	–	–	–
Transformer (scratch)	–	–	–	–	–	–
Transformer (pretrained)	–	–	–	–	–	–
DGCNN [34]	1.05/11.7	51.5/18.1	51.3/15.2	0.11/1.04	89.1/27.7	89.1/25.4
			Persistence-based models			
PersFormer [29]	–	–	–	–	–	–
xPerT [22]	–	–	–	–	–	–
PersLay [5]	–	–	–	–	–	–

Table 2: **Performance of baseline models trained on DONUT and evaluated both in-distribution and on ABC.** Values are reported as *DONUT/ABC*. We report mean squared error (MSE), balanced accuracy (Acc.), and balanced F1-score (F1). MSE is reported to capture how far off predictions are from the ground-truth on average, which is particularly relevant given the natural hierarchy of the labels (genus and connected components): misclassifying by a larger margin is more severe than a closer miss.

2.3.3 Saliency Analysis

To further understand model behavior, we visualize saliency maps on toy shapes. These maps highlight which regions contribute most to predictions. When trained to predict genus, models tend to focus on cycles, while for connected components they concentrate around contact points between shapes (Figure **??**). Such results suggest that models attend to meaningful topological structures rather than exploiting geometric shortcuts.



(a) Label distributions for genus and connected components.

(b) Scalability of dataset generation.

Figure 7: **General properties of DONUT.** *(a),(b)* The marginal distributions of genus and connected components are approximately uniform, ensuring balanced supervision across topological classes. *(c)* The time required to generate datasets of varying sizes demonstrates that even large-scale variants remain practical.

2.4 Limitations and Further Improvements

Topological Variety. The current dataset has limited topological variety. It only encodes two characteristics: genus and number of connected components. This is mainly because all meshes are watertight and manifold. In this setting, topological invariants are trivial to deduce ($\beta_1 = 2 \times (\text{"}genus\text{"})$, $\beta_2 = \beta_0$). A more challenging extension would include surfaces with boundaries, non-watertight meshes, or nested shapes. While the current generation process allows nested components in theory, no such cases are present in practice. Another direction is to incorporate families of parametric shapes with non-trivial topology, such as catenoids, helicoids, or Möbius strips.

Geometric Complexity. The geometric diversity is also limited. It could be extended by adding parametric families that introduce richer geometric structures. For example, the Superformula introduced by Gielis (2003) [13] generalizes superellipses and provides a broad range of shapes. Although we did not add such families here, since the dataset is mainly used for evaluation and no overfitting issues were observed with the current configuration, this would be valuable in other settings. In particular, if the dataset is used for pretraining, as in Point-MAE-Zero, introducing more geometric complexity would improve its utility.

3 Used Models

This section provides an extensive description of the models used in our experiments. While most of them are transformer-based, they all come with key architectural variations that need to be explained to understand the obtained results. What's more, none of the baselines are transformer-based. Therefore, having a fair comparison between models may not be relevant at first glance. That's why we highlight the key differences and similarities between the models MISSING SMTH....

3.1 Baselines Models

3.2 Transformer-based models

Recent advances in transformer architectures for 3D shape understanding have largely built on the intuition that ideas from NLP and vision can be transferred to point cloud data. While the geometry of 3D shapes is fundamentally different from text or images, most models share a similar pipeline. A point cloud is first partitioned into a set of local patches. This is commonly achieved with farthest point sampling, and then grouping each center with its nearest neighbors to form a subcloud. Each of these patches is then embedded into a high-dimensional space using a lightweight encoder, typically a variant of PointNet or DGCNN that aggregates local geometric features while maintaining permutation invariance.

To preserve spatial structure, positional encodings are added by projecting the coordinates of the patch centers through an MLP, which provides the transformer with a notion of relative geometry between patches. The resulting sequence of patch embeddings, optionally augmented with a learnable global "class token" is

then fed into a Transformer encoder. This process is conceptually identical to Vision Transformers, where image patches are tokenized, embedded, and processed sequentially, though here the tokens correspond to irregular geometric neighborhoods rather than regular image patches.

This generic framework has provided the foundation for several influential works that differ primarily in how tokens are defined, how self-supervised pretraining is formulated, and whether hierarchical representations are incorporated. We now discuss these paradigms in detail.

3.2.1 Point-BERT: Discrete Tokenization and Masked Modeling

One of the earliest attempts to port the BERT paradigm to point clouds is Point-BERT [36]. Its central idea is to treat a point cloud as a sequence of discrete tokens, thereby enabling the use of masked language modeling (MLM) for self-supervised pretraining. To achieve this, Point-BERT first trains a discrete variational autoencoder (dVAE) to learn a vocabulary of geometric tokens. The encoder of the dVAE maps continuous patch embeddings into discrete codewords, while the decoder reconstructs the original patch from the codeword, ensuring that the learned vocabulary captures meaningful local geometry.

With this vocabulary in place, Point-BERT pretrains the Transformer using Masked Point Modeling (MPM): a subset of input patches are masked, and the model is trained to predict the corresponding discrete token indices assigned by the dVAE. This formulation parallels masked language modeling in NLP, where the model predicts missing words from context, but here the "words" are geometric patches. Importantly, Point-BERT also incorporates a learnable class token that is updated jointly with the patch tokens during pretraining. The final embedding of this class token is then used for downstream classification and retrieval tasks, mirroring the global semantics captured by the CLS token in NLP transformers.

This approach makes pretraining strongly analogous to BERT, but it requires a two-stage pipeline, first learning the tokenizer and then training the transformer, which introduces additional complexity. Nonetheless, Point-BERT demonstrated that masked token prediction is a powerful way to learn shape priors from unlabeled point clouds.

3.2.2 Point-MAE: Continuous Tokens and Masked Autoencoding

While Point-BERT relies on discrete tokenization, Point-MAE [26] instead follows the Masked Autoencoder (MAE) [37] framework introduced in vision. Here, patches are represented by continuous embeddings rather than discrete vocabulary entries, and the pretraining task is reconstruction rather than classification. Specifically, Point-MAE applies a high masking ratio (often exceeding 60%) to the input patches. Only the unmasked patches are fed through the heavy transformer encoder. Masked patches are replaced with a shared mask token, and both unmasked embeddings and mask tokens are processed by a lightweight decoder tasked with reconstructing the original 3D coordinates of the masked patches.

This approach has several advantages. First, it avoids the need for a separate tokenizer, simplifying the training pipeline. Second, it is more computationally efficient: since the encoder processes only unmasked patches, the model scales better to large point clouds. Third, it allows the model to directly learn continuous geometric representations, which are more flexible than a finite vocabulary.

Unlike Point-BERT, Point-MAE does not include a CLS token during pretraining. Instead, the class token is introduced only when the model is fine-tuned on downstream tasks. This choice emphasizes that the focus of pretraining is purely local reconstruction, while global semantics can be learned in task-specific adaptation. Overall, Point-MAE demonstrated that masked autoencoding provides a simple yet effective way to pretrain 3D transformers, establishing a new standard for self-supervised learning on point clouds.

3.2.3 Hierarchical Extensions: Multi-Scale Representations

A third line of work emphasizes that 3D shapes are inherently hierarchical, with geometric structures manifesting at multiple scales. This insight parallels the evolution from PointNet to PointNet++, where hierarchical feature extraction proved essential for robust representation learning.

Point-M2AE [39] extends Point-MAE by introducing a multi-scale transformer architecture. The encoder progressively downsamples the point cloud into coarser patches, capturing increasingly global features, while

	ModelNet				DONUT			
	5-way		10-way		5-way		10-way	
Methods	10-shot	20-shot	10-shot	20-shot	10-shot	20-shot	10-shot	20-shot
Point-BERT [36]	$94.6_{\pm 3.1}$	$96.3_{\pm 2.7}$	$91.0_{\pm 5.4}$	$92.7_{\pm 5.1}$	–	–	–	–
Point-BERT (ABC)	$95.9_{\pm 2.1}$	$98.1_{\pm 1.8}$	$91.7_{\pm 4.5}$	$94.2_{\pm 3.4}$	–	–	–	–
Point-MAE [26]	$96.3_{\pm 2.5}$	$97.8_{\pm 1.8}$	$92.6_{\pm 4.1}$	$95.0_{\pm 3.0}$	$32.9_{\pm 3.5}$	$38.1_{\pm 3.9}$	$20.4_{\pm 2.1}$	$23.8_{\pm 1.7}$
Point-MAE (ABC)	$96.2_{\pm 2.7}$	$97.6_{\pm 1.9}$	$92.3_{\pm 4.4}$	$95.2_{\pm 2.9}$	$35.1_{\pm 6.0}$	$40.1_{\pm 4.9}$	$21.4_{\pm 2.7}$	$22.8_{\pm 2.4}$
PCP-MAE [40]	$97.4_{\pm 2.3}$	$99.1_{\pm 0.8}$	$93.5_{\pm 3.7}$	$95.9_{\pm 2.7}$	–	–	–	–
PCP-MAE (ABC)	$96.1_{\pm 3.1}$	$98.3_{\pm 4.3}$	$91.7_{\pm 4.3}$	$94.9_{\pm 2.9}$	–	–	–	–
PM2AE [39]	$97.4_{\pm 2.3}$	$99.1_{\pm 0.8}$	$93.5_{\pm 3.7}$	$95.9_{\pm 2.7}$	–	–	–	–
PM2AE (ABC)	$96.1_{\pm 3.1}$	$98.3_{\pm 4.3}$	$91.7_{\pm 4.3}$	$94.9_{\pm 2.9}$	–	–	–	–

Table 3: **Few-shot object classification on ModelNet40.** We conduct 10 independent experiments for each setting and report mean accuracy (%) with standard deviation.

the decoder reconstructs masked patches using skip connections, reminiscent of U-Net architectures. In addition, Point-M2AE employs a multi-scale masking strategy to ensure consistency across levels of abstraction and introduces local spatial self-attention at fine scales to better capture detailed geometry. This design not only improves parameter efficiency but also strengthens the model's ability to capture structures across varying spatial resolutions.

3.2.4 Further Improvements

Most recent approaches have mainly built on top of Point-MAE, primarily for the training setting, easier than Point-BERT. PCP-MAE [40] modifies positional encoding to prevent information leakage: instead of directly using patch centers, which can trivially reveal masked locations, it requires the model to predict them. HFBRI-MAE [35] incorporates handcrafted rotation-invariant features as both tokens and positional encodings, improving robustness under arbitrary 3D rotations. Point-RAE [24] proposes a regression-before-autoencoding strategy to decouple encoder and decoder representations, thereby avoiding the risk of the decoder dominating feature learning. Collectively, these hierarchical and refined models show that injecting geometric inductive biases into the transformer architecture can significantly improve generalization and robustness.

3.2.5 Comparative View

Despite their differences, Point-BERT, Point-MAE, and their hierarchical successors share a common foundation: the adaptation of the transformer paradigm from language and vision to point clouds through *patchification*, embedding, and positional encoding. What distinguishes them is how tokens are defined, how the self-supervised objective is formulated, and whether hierarchical geometry is explicitly modeled. Discrete-token approaches like Point-BERT require a two-stage pretraining pipeline but tightly align with NLP pretraining paradigms. Continuous-token methods like Point-MAE are simpler and more efficient, emphasizing reconstruction over classification. Hierarchical extensions such as Point-M2AE incorporate multiscale representations, bringing additional inductive structure into the transformer.

An important consequence of these differences is that evaluation protocols vary across models. Point-BERT pretraining relies on predicting discrete token IDs, while Point-MAE and M2AE use reconstruction losses such as Chamfer distance. The role of the CLS token also differs, being central during pretraining in Point-BERT but absent in MAE-style approaches. As a result, downstream fine-tuning and linear probing must be carefully adapted to each architecture. The precise evaluation settings for each method will be detailed in the following experimental section.

4 Subspace Alignment with Persistent Homology

We study the relationship between representations learned by 3D shape encoders and those derived from persistent homology. Following the perspective of the Platonic Representation Hypothesis [15, 20], we view learned embeddings as potentially containing subspaces that capture intrinsic structural information. Information captured by persistent homology can also be treated as a complementary modality. Assuming there's a way to encode topological features into a vector space, we can then ask whether the learned and topological subspaces align. It's worth noticing that the analogy with [20] is not perfect: while it's assumed that representations tend to converge at scale, topological features, even learned ones, aren't the result of large scale training.

We first review background on persistent homology and its vectorization. We then describe our alignment protocol and present results across encoders and datasets.

4.1 Persistence Diagrams

Filtrations. Let $(X, \leq)$ be a topological space equipped with a real index. A (real) filtration is a family of subspaces $(X_a)_{a\in\mathbb{R}}$ with $X_a \subseteq X_b$ whenever $a \leq b$ and $\bigcup_a X_a = X$. Typical examples are sublevel sets of a function $f : X \to \mathbb{R}$, namely $X_a = f^{-1}((-\infty, a])$.

Homology and persistent maps. Fix a field $\Bbbk$ and compute singular or simplicial homology with coefficients in $\Bbbk$. For $k \geq 0$ and $a \leq b$, the inclusion $X_a \hookrightarrow X_b$ induces a linear map

$$i_a^b : H_k(X_a; \Bbbk) \longrightarrow H_k(X_b; \Bbbk). \tag{6}$$

A k-dimensional class $\alpha \in H_k(X_b)$ is *born* at the smallest a for which it has a representative in $H_k(X_a)$, and it *dies* at the smallest $d > b$ where it maps to zero in $H_k(X_d)$ under i_b^d. Classes that never die are called *essential*.

Persistence modules, barcodes and diagrams. The collection $\{H_k(X_a), i_a^b\}_{a\leq b}$ is a persistence module. Under standard finiteness conditions (e.g. f tame, or finite type filtrations), it decomposes into interval modules. This gives a multiset of intervals (the barcode) or equivalently a multiset of points (b_i, d_i) in the open half plane $\{(x, y) \in \mathbb{R}^2 : x < y\}$ together with the diagonal $\{(t, t)\}$ available for matching with infinite multiplicity. This multiset is the k-th persistence diagram $\mathrm{Dgm}_k(X)$.

Why a persistence diagram is not a Euclidean vector object. A persistence diagram is a locally finite *multiset* in $\mathbb{R}^2$ with a special diagonal of infinite multiplicity. There is no canonical addition or scalar multiplication that preserves the intended topology. Barycenters can be non unique. Diagrams can have countably many off diagonal points if tameness fails. Hence one treats diagrams as elements of a metric space rather than a linear space.

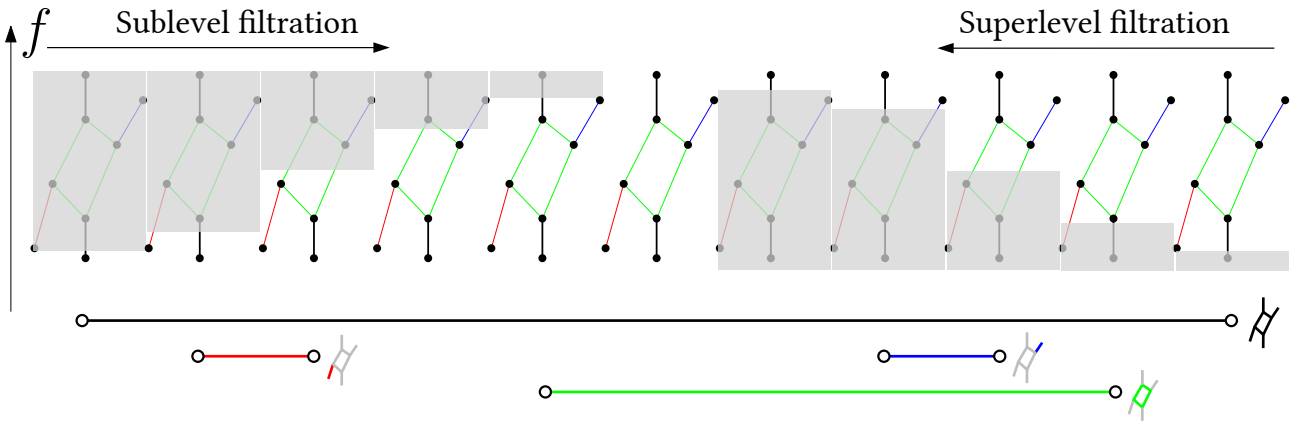
Distances between diagrams. Let D and E be diagrams. We allow matchings to the diagonal. The *bottleneck distance* is

$$d_B(D, E) \;=\; \inf_{\gamma: D\to E} \sup_{x\in D} \|x - \gamma(x)\|_\infty. \tag{7}$$

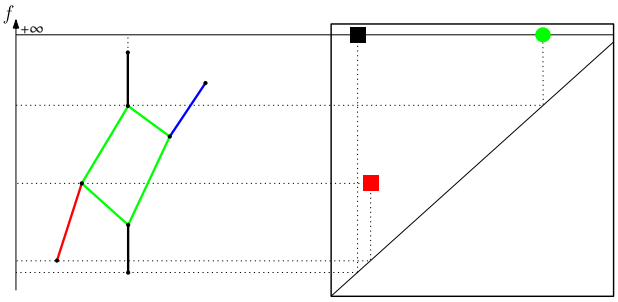
For $1 \leq p < \infty$, the *p-Wasserstein distance* is

$$d_{W,p}(D, E) \;=\; \left(\inf_{\gamma: D\to E} \sum_{x\in D} \|x - \gamma(x)\|_\infty^p \right)^{1/p}. \tag{8}$$

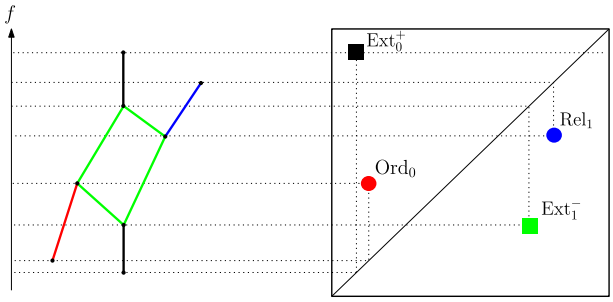
These are the standard and stable distances on persistence diagrams. The plain Hausdorff distance between the underlying point sets does not account for multiplicities and diagonal matchings and is not used in modern stability theory.



(a) Sub- (resp. super-) level set filtration.



(b) Left panel explanation.



(c) Right panel explanation.

Figure 8: **Ordinary vs extended persistence.** (Figures adapted from [5]). In 8a, each node of the graph is assigned its height. Persistence intervals are shown under the sequence. In 8b, ordinary persistence captures connected components and loops, but essential classes lead to infinite intervals (black and green markers) and the upward branch (blue) isn't captured. In 8c, extended persistence pairs all classes at finite values by combining sublevel and superlevel filtrations with relative homology.

Stability theorems. For sublevel set filtrations of functions $f, g : X \to \mathbb{R}$ on a common triangulable space, the diagrams are stable under perturbations of f:

$$d_B\big(\mathrm{Dgm}_k(f), \mathrm{Dgm}_k(g)\big) \;\le\; \|f - g\|_\infty. \tag{9}$$

Related bounds hold for $d_{W,p}$. In the algebraic language of persistence modules, the Isometry Theorem states that for q-tame modules M, N,

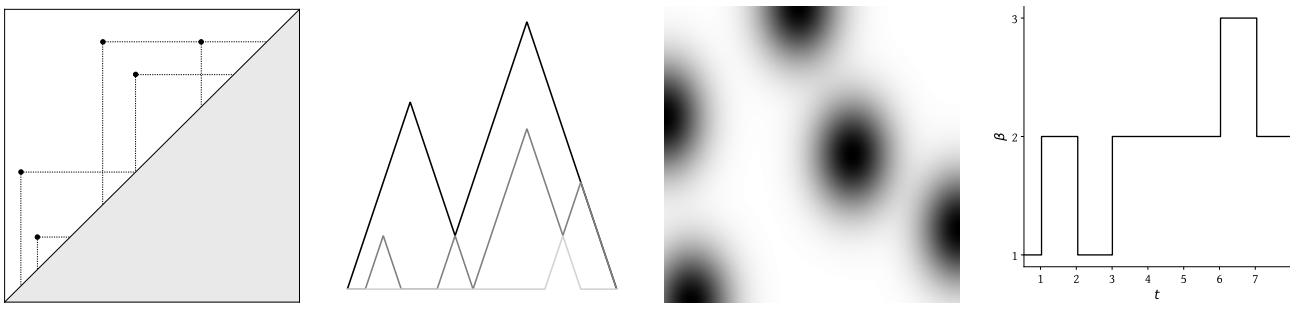
$$d_B\big(\mathrm{Dgm}(M), \mathrm{Dgm}(N)\big) \;=\; d_I(M, N), \tag{10}$$

where d_I is the interleaving distance.

Extended persistence. Essential classes can lead to pairs with infinite lifetime. Extended persistence remedies this by combining sublevel and superlevel filtrations and using relative homology. For a function $f : X \to \mathbb{R}$ on a compact space, consider the sublevel filtration (X_a) and the superlevel filtration $(X^a = f^{-1}([a, \infty)))$. One constructs a zigzag that passes from absolute to relative homology so that every class is paired at a finite parameter value. The resulting *extended persistence diagram* has only finite pairs and encodes essential features as finite points.

4.2 Vectorization of Persistence Diagrams

Vectorizing persistence diagrams is crucial for incorporating topological signals in standard learning pipelines. A persistence diagram is a multiset of points in the plane. It does not live in a vector space and is usually compared with bottleneck or Wasserstein distances rather than inner products. Most learning algorithms expect fixed size vectors or a Hilbert space structure, so raw diagrams are not a natural input. Diagrams also



(a) Persistence Diagram (b) Persistence Landscape (c) Persistence Image (d) Betti Curve

Figure 9: **Common prescribed vectorizations of a persistence diagram.** From left to right: (a) A sample persistence diagram with points representing topological features; (b) Persistence landscape capturing the prominence of features across scales; (c) Persistence image providing a smoothed, grid-based representation; *Note:* Instead of considering persistence pairs as a *(birth,death)* tuple, persistence image is fitted on $(birth, death - birth)$ pairs. (d) Betti curve showing the count of features over filtration values.

have variable size and are permutation invariant, which complicates batching and optimization. These issues motivate stable feature maps and kernels that embed diagrams into Euclidean

Prescribed Vectorization. Handcrafted summaries have long been the dominant strategy for vectorizing persistence diagrams in machine learning. The most common approaches include persistence landscapes [4], Betti curves [14], and persistence images [1] (Figure 9). Landscapes embed diagrams in a Hilbert space and enable classical statistics, but they still require design choices such as the number of landscape levels and sampling resolution. Betti curves collapse a diagram to simple counting functions over the filtration. Persistence images rasterize diagrams with a kernel on a fixed grid. These methods are stable in theory but depend on key hyperparameters such as grid resolution and kernel width. These choices must be tuned per task and can limit expressiveness and generalization. Finally, ATOL [30] proposes an unsupervised alternative. It vectorizes measures, including sets of persistence points, by first clustering with K-means and then summarizing cluster statistics in a fixed length descriptor. The pipeline is simple and fast but remains sensitive to the chosen number of clusters and to the distribution shift between training and test diagrams.

Learned Vectorization. Recent work has moved from handcrafted features to learning based approaches [25],[16] that act on persistence diagrams more directly, often through extended persistence [8] (see Section ...). PersLay [5] introduces a neural layer that learns stable weights and pooling functions over points in a diagram. It was developed for graph tasks and relies on extended persistence to encode informative signatures before the learned aggregation. PLLay [21] follows a hybrid route. It first computes persistence landscapes and then applies a differentiable neural layer that learns how to weight and combine landscape coordinates. This reduces manual feature design while keeping the stability benefits of landscapes. Transformer based designs have also appeared. Persformer [29] treats each persistence point as a token and applies standard self attention. This is flexible but scales poorly as the number of points grows. To address scalability, the Multiset Transformer [32] clusters points and modifies attention to account for multiplicities, so that a cluster can be handled as a single item with weight greater than one. This change reduces memory and time while preserving permutation invariance at the multiset level. xPerT [22] goes further by leveraging sparsity with a *pixelized diagram* representation. It is close in spirit to persistence images but does not require a smoothing kernel. The pixeliaation allows efficient tokenization and improves scalability compared with Persformer. xPerT also works with extended persistence; details are deferred to the dedicated section.

Across these learning based methods, a common trait is that the diagram is manipulated explicitly rather than replaced by coarse handcrafted summaries. Most results to date are on graph benchmarks rather than 3D shape understanding. This gap limits conclusions for point cloud encoders trained in a self supervised way, where task independent evaluation is required.

Finally, recent topology aware 3D generation pipelines [17] directly encode persistence pairs. More precisely they feed the top k most persistent pairs as $\{g_i = (b_i, d_i - b_i)\}_{i \in [\![1,k]\!]}$ where b_i is the birth time and d_i is the death time of the i-th pair. These papers report increased diversity or topology control but are typically demonstrated on a few ShapeNet categories or related datasets. A representative example conditions a latent diffusion model on topological features computed from persistence diagrams and validates on ShapeNet and

ABC.

4.3 Subspace Alignment Protocol

A Supplementary Material for DONUT

This section first provides an in-depth analysis of the EuLearn dataset [12], to justify why it's not reliable for our experiments. Then, it details the algorithms and proofs related to the DONUT dataset generation process.

A.1 EuLearn Dataset Analysis

A key issue is the entanglement between topology and orientation. As shown in Figure 11, a simple k-nearest neighbor classifier trained on point clouds in their canonical orientation achieves near-perfect accuracy. However, accuracy collapses when test samples are randomly rotated, despite genus being a topological invariant. This indicates that classifiers exploit orientation-dependent cues instead of genuine topological structure.

The same effect appears at the representation level. Figure 10 shows UMAP projections of Point-BERT embeddings. When evaluated on canonical orientations, embeddings form tight clusters by genus. After applying random rotations, this structure disappears entirely. If embeddings reflected topology, genus separation should persist under rigid transformations; instead, the disappearance of clusters confirms that the genus signal in EuLearn is largely a proxy for global pose and correlated geometric features.

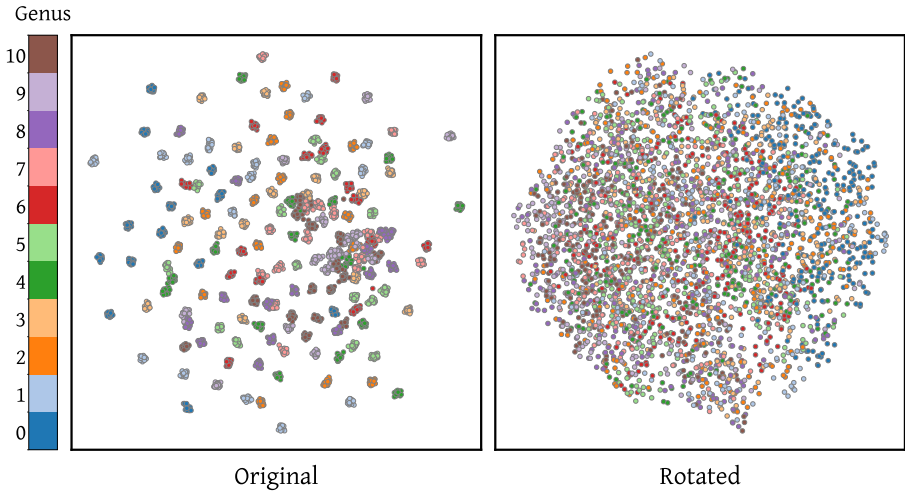


Figure 10: **UMAP projections of Point-BERT embeddings colored by genus.** Clear clusters by genus appear for canonical orientations (left) but vanish under random rotations (right), showing that genus separation in EuLearn is tied to orientation rather than topology.

Linear probing experiments further reinforce this point (Table 12). Probes trained on canonical embeddings generalize only to canonical test shapes, collapsing under rotations. Probes trained on rotated embeddings avoid orientation bias but fail to recover meaningful genus structure, with accuracies barely above random chance. In both cases, the embeddings overfit to spurious correlations, not topology.

Taken together, these results demonstrate that EuLearn's construction introduces systematic biases. The dataset does not allow disentangling true topological reasoning from pose-dependent shortcuts, making it unsuitable for evaluating structural understanding in 3D shape encoders.

A.2 Sampling Labels

As mentioned in Section 2.2.1, labels (number of connected components β_0 and genus g) are sampled prior to generating any mesh. This step is key as it is where we ensure marginal distributions of both labels to be approximately uniform. The sampling procedure is described below. The algorithm takes as input the minimum and maximum number of connected components β_0^{min} and β_0^{max}, the maximum genus per component g^{max}, the maximum genus per sample (group of components) G^{max} and a parameter k that controls the number of samples per value of β_0. It outputs a list of (β_0, g) pairs, with $\beta_0 \sim \mathcal{U}[\![\beta_0^{min}, \beta_0^{max}]\!]$ and $g \sim \mathcal{U}[\![0, G^{max}]\!]$ subject to the constraint that $g \leq \min(G^{max}, \beta_0 \times g^{max})$. The constraint encodes that a single component can't have a genus higher than g^{max}, and that the total genus of a sample cannot exceed G^{max}.

Figure 11: **Classification accuracy of a k-nearest neighbor classifier vs. training rotation range.** Accuracy is perfect when training and testing on canonical orientations but collapses as random rotations along the z-axis are introduced, revealing orientation-dependent bias in EuLearn. *Note:* Results are cross-validated with 5 folds.

Training set	Test set	Training Acc.	Test Acc.
⊠	⊠	99.9	$96.4_{(-0.5)}$
	↻		$16.3_{(-83.6)}$
↻	↻	77.8	$50.1_{(-27.7)}$
	⊠		$45.9_{(-31.9)}$

Figure 12: **Linear probing accuracy on Point-BERT embeddings of canonical vs. rotated shapes.** Probes trained on canonical orientations (⊠) achieve high accuracy only on canonical test data, collapsing under rotation. (↻). Probes trained on rotated shapes generalize poorly in both cases, confirming that EuLearn does not provide a stable signal for genus.

Role of k. Choosing exactly k times each value of β_0 ensures that the marginal distribution of β_0 is perfectly uniform. However, since the number of valid genus values depends on β_0, the marginal distribution of g is only approximately uniform.

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

let $\mathfrak{B}_0 \leftarrow \{\underbrace{\beta_0^{min},\dots,\beta_0^{min}}_{\times k}, \underbrace{\beta_0^{min}+1}_{\times k},\dots,\underbrace{\beta_0^{max}}_{\times k}\}$
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

A.3 Choosing Components Shapes

Once the labels are sampled, the next step is to choose the actual shapes composing each sample. This is done by decomposing the global genus g into β_0 individual genera $(g_1, \dots, g_{\beta_0})$ such that $\sum_{i=1}^{\beta_0} g_i = g$. Each g_i is then associated with a parametric family of shapes (Section 2.2.2). However, given a couple (β_0, g) several shapes configurations can lead to the same global labels (Figure ...). We leverage this observation to add more variability within the dataset, by first getting all the possible combinations of shapes for a given label configuration. The problem can be formulated as follows:

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

Table 4: Hyperparameter values used for the DONUT dataset for the experiments. Probing tasks boil down to classifying shapes across $\beta_0^{\max} - \beta_0^{\min} + 1 = 6$ categories for connected components and 11 categories for genera.

Problem statement: *We're given three integers: a (total items), b (target weighted sum) and g^{max} (maximum template index). We also have templates numbered from 0 to g^{max} (included). Template i has a weight of i.*
We want to find all the possible ways to distribute exactly a items across these templates such that the weighted sum $\sum_{i=0}^{g^{max}} i \cdot x_i = b$, where x_i is the number of items assigned to template i.
Input:

- a: Total number of items to distribute
- b: Target weighted sum
- g^{max}: Maximum template index (templates are $0, 1, 2, \ldots, g^{max}$)

Output: A list of all possible distributions $(x_0, x_1, \ldots, x_{g^{max}})$.

Example:
Input: $a = 3,\ b = 5,\ g^{max} = 3$
Output: $\big[[0, 2, 0, 1], [1, 0, 1, 1], [0, 1, 2, 0]\big]$
Explanation:

- $[0, 2, 0, 1]$: $0 \cdot 0 + 2 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 = 3$, and total count $0 + 2 + 0 + 1 = 3$
- $[1, 0, 1, 1]$: $1 \cdot 0 + 0 \cdot 1 + 1 \cdot 2 + 1 \cdot 3 = 5$, and total count $1 + 0 + 1 + 1 = 3$
- $[0, 1, 2, 0]$: $0 \cdot 0 + 1 \cdot 1 + 2 \cdot 2 + 0 \cdot 3 = 5$, and total count $0 + 1 + 2 + 0 = 3$

In the context of DONUT, a corresponds to the number of connected components β_0, b to the total genus g and g^{max} to the maximum genus per component. The output distributions $(x_0, x_1, \ldots, x_{g^{max}})$ indicate how many components of each genus should be used to form a sample with the desired labels. For instance, if $\beta_0 = 3$, $g = 5$ and $g^{max} = 3$, one valid output is $(1, 0, 1, 1)$ which means that the sample should be composed of one genus-0 shape, zero genus-1 shapes, one genus-2 shape and one genus-3 shape. *Note* that G^{max} isn't directly involved here, as it's been used already to choose the values of g (Algorithm 1).

Algorithmic Details. Even though we seek for efficient algorithms to ensure the scalability of DONUT, finding all the combinations requires a tree search algorithm with backtracking. This procedure has exponential () complexity. However, since both β_0^{max} and G^{max} are small, enumerating all the solutions given (β_0, g) requires a negligible amount of time compared to generating the actual meshes and composing them together. Once all the solutions are enumerated, we randomly sample one of them to generate the actual meshes. *Note* that to avoid useless computations, we cache the solutions for each (β_0, g) couple, so that if the same couple is encountered again, we can directly sample from the already computed solutions.

Complexity. A naïve approach of the algorithm's complexity is $O(a^{g^{max}+1})$, since at each of the $g^{max} + 1$ recursion levels, we can have up to a branches. However, this bound is very loose, because the additional

constraint on the weighted sum b prunes most fo the branches: many partial assignments can't possbbly satisfy both the total count and the target sum, and are therefore discarded early. The algorithm actually enumerates *weak combinations* [31]

Algorithm 2 ENUMERATE-SOLUTIONS
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
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

A.4 Baselines Training Details

We provide here the training details for the baselines presented in Section 2.

	Point-based models			Topology-based models		
Hyperparameter	PointNet	PointNet++	DGCNN	PersFormer	xPerT	PersLay
Batch size	32	32	32	64	–	–
Learning rate	0.001	0.001	0.005	0.0005	–	–
Optimizer	Adam	Adam	Adam	AdamW	–	–
Weight decay	1e-4	1e-4	1e-4	5e-5	–	–
Dropout	0.3	0.3	0.5	0.5	–	–
Training epochs	200	200	250	300	–	–

Table 5: Comparison of typical training hyperparameters for point-based models (PointNet, PointNet++, DGCNN) and topology-based models (PersFormer, xPerT, PersLay). A vertical line separates the two families of architectures.

A.5 Additional Results

We complete the results provided in Table 2 with per-class metrics for each baseline.

B Supplementary Definitions

B.1 Metrics

Balanced accuracy For a classification task with K classes, the balanced accuracy is defined as:

$$\text{Balanced Accuracy} = \frac{1}{K}\sum_{i=1}^{K}\frac{TP_i}{TP_i + FN_i} \tag{11}$$

where TP_i and FN_i are the number of true positives and false negatives for class i, respectively. This metric is particularly useful for imbalanced datasets, as each class is reweighted according to its importance.

References

[1] Henry Adams et al. *Persistence Images: A Stable Vector Representation of Persistent Homology*. 2016. arXiv: 1507.06217 [cs.CG]. URL: https://arxiv.org/abs/1507.06217.

[2] Rubén Ballester et al. *MANTRA: The Manifold Triangulations Assemblage*. 2025. arXiv: 2410.02392 [cs.LG]. URL: https://arxiv.org/abs/2410.02392.

[3] Alan H. Barr. "Superquadrics and Angle-Preserving Transformations". In: *IEEE Computer Graphics and Applications* 1.1 (1981), pp. 11–23. DOI: 10.1109/MCG.1981.1673799.

[4] Peter Bubenik. *Statistical topological data analysis using persistence landscapes*. 2015. arXiv: 1207.6437 [math.AT]. URL: https://arxiv.org/abs/1207.6437.

[5] Mathieu Carrière et al. *PersLay: A Neural Network Layer for Persistence Diagrams and New Graph Topological Signatures*. 2020. arXiv: 1904.09378 [stat.ML]. URL: https://arxiv.org/abs/1904.09378.

[6] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. arXiv: 1512.03012 [cs.GR]. URL: https://arxiv.org/abs/1512.03012.

[7] Xuweiyi Chen and Zezhou Cheng. *Learning 3D Representations from Procedural 3D Programs*. 2025. arXiv: 2411.17467 [cs.CV]. URL: https://arxiv.org/abs/2411.17467.

[8] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. "Extending Persistence Using Poincaré and Lefschetz Duality". In: *Foundations of Computational Mathematics* 9.1 (2009), pp. 79–103. DOI: 10.1007/s10208-008-9027-z.

[9] Matt Deitke et al. *Objaverse: A Universe of Annotated 3D Objects*. 2022. arXiv: 2212.08051 [cs.CV]. URL: https://arxiv.org/abs/2212.08051.

[10] Matt Deitke et al. *Objaverse-XL: A Universe of 10M+ 3D Objects*. 2023. arXiv: 2307.05663 [cs.CV]. URL: https://arxiv.org/abs/2307.05663.

[11] Elisabetta Fedele et al. *SuperDec: 3D Scene Decomposition with Superquadric Primitives*. 2025. arXiv: 2504.00992 [cs.CV]. URL: https://arxiv.org/abs/2504.00992.

[12] Rodrigo Fritz et al. *EuLearn: A 3D database for learning Euler characteristics*. 2025. arXiv: 2505.13539 [cs.CG]. URL: https://arxiv.org/abs/2505.13539.

[13] Johan Gielis. "A generic geometric transformation that unifies a wide range of natural and abstract shapes". In: *American Journal of Botany* 90.3 (2003), pp. 333–338. DOI: 10.3732/ajb.90.3.333.

[14] Chad Giusti et al. "Clique topology reveals intrinsic geometric structure in neural correlations". In: *Proceedings of the National Academy of Sciences* 112.44 (Oct. 2015), 13455–13460. ISSN: 1091-6490. DOI: 10.1073/pnas.1506407112. URL: http://dx.doi.org/10.1073/pnas.1506407112.

[15] Souhail Hadgi et al. *Escaping Plato's Cave: Towards the Alignment of 3D and Text Latent Spaces*. 2025. arXiv: 2503.05283 [cs.CV]. URL: https://arxiv.org/abs/2503.05283.

[16] Christoph Hofer et al. *Deep Learning with Topological Signatures*. 2018. arXiv: 1707.04041 [cs.CV]. URL: https://arxiv.org/abs/1707.04041.

[17] Jiangbei Hu et al. *Topology-Aware Latent Diffusion for 3D Shape Generation*. 2024. arXiv: 2401.17603 [cs.CV]. URL: https://arxiv.org/abs/2401.17603.

[18] Jingwei Huang, Hao Su, and Leonidas Guibas. *Robust Watertight Manifold Surface Generation Method for ShapeNet Models*. 2018. arXiv: 1802.01698 [cs.CG]. URL: https://arxiv.org/abs/1802.01698.

[19] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. *ManifoldPlus: A Robust and Scalable Watertight Manifold Surface Generation Method for Triangle Soups*. 2020. arXiv: 2005.11621 [cs.GR]. URL: https://arxiv.org/abs/2005.11621.

[20] Minyoung Huh et al. *The Platonic Representation Hypothesis*. 2024. arXiv: 2405.07987 [cs.LG]. URL: https://arxiv.org/abs/2405.07987.

[21] Kwangho Kim et al. *PLLay: Efficient Topological Layer based on Persistence Landscapes*. 2021. arXiv: 2002.02778 [cs.LG]. URL: https://arxiv.org/abs/2002.02778.

[22] Sehun Kim. *xPerT: Extended Persistence Transformer*. 2024. arXiv: 2410.14193 [cs.LG]. URL: https://arxiv.org/abs/2410.14193.

[23] Sebastian Koch et al. *ABC: A Big CAD Model Dataset For Geometric Deep Learning*. 2019. arXiv: 1812.06216 [cs.GR]. URL: https://arxiv.org/abs/1812.06216.

[24] Yang Liu et al. "Regress Before Construct: Regress Autoencoder for Point Cloud Self-supervised Learning". In: *Proceedings of the 31st ACM International Conference on Multimedia*. MM '23. ACM, Oct. 2023, 1738–1749. DOI: 10.1145/3581783.3612106. URL: http://dx.doi.org/10.1145/3581783.3612106.

[25] Naoki Nishikawa, Yuichi Ike, and Kenji Yamanishi. *Adaptive Topological Feature via Persistent Homology: Filtration Learning for Point Clouds*. 2023. arXiv: 2307.09259 [cs.LG]. URL: https://arxiv.org/abs/2307.09259.

[26] Yatian Pang et al. *Masked Autoencoders for Point Cloud Self-supervised Learning*. 2022. arXiv: 2203.06604 [cs.CV]. URL: https://arxiv.org/abs/2203.06604.

[27] Charles R. Qi et al. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV]. URL: https://arxiv.org/abs/1706.02413.

[28] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV]. URL: https://arxiv.org/abs/1612.00593.

[29] Raphael Reinauer, Matteo Caorsi, and Nicolas Berkouk. *Persformer: A Transformer Architecture for Topological Machine Learning*. 2022. arXiv: 2112.15210 [cs.LG]. URL: https://arxiv.org/abs/2112.15210.

[30] Martin Royer et al. *ATOL: Measure Vectorization for Automatic Topologically-Oriented Learning*. 2020. arXiv: 1909.13472 [cs.CG]. URL: https://arxiv.org/abs/1909.13472.

[31] Bruce E. Sagan. *Combinatorics: The Art of Counting*. Vol. 210. Graduate Studies in Mathematics. American Mathematical Society, 2020, p. 304. ISBN: 978-1-4704-6032-7.

[32] Minghua Wang, Ziyun Huang, and Jinhui Xu. *Multiset Transformer: Advancing Representation Learning in Persistence Diagrams*. 2024. arXiv: 2411.14662 [cs.LG]. URL: https://arxiv.org/abs/2411.14662.

[33] Peng-Shuai Wang, Yang Liu, and Xin Tong. "Dual Octree Graph Networks for Learning Adaptive Volumetric Shape Representations". In: *ACM Transactions on Graphics (SIGGRAPH)* 41.4 (2022).

[34] Yue Wang et al. *Dynamic Graph CNN for Learning on Point Clouds*. 2019. arXiv: 1801.07829 [cs.CV]. URL: https://arxiv.org/abs/1801.07829.

[35] Xuanhua Yin et al. *HFBRI-MAE: Handcrafted Feature Based Rotation-Invariant Masked Autoencoder for 3D Point Cloud Analysis*. 2025. arXiv: 2504.14132 [cs.CV]. URL: https://arxiv.org/abs/2504.14132.

[36] Xumin Yu et al. *Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling*. 2022. arXiv: 2111.14819 [cs.CV]. URL: https://arxiv.org/abs/2111.14819.

[37] Yaohua Zha et al. *Towards Compact 3D Representations via Point Feature Enhancement Masked Autoencoders*. 2023. arXiv: 2312.10726 [cs.CV]. URL: https://arxiv.org/abs/2312.10726.

[38] Longwen Zhang et al. *CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets*. 2024. arXiv: 2406.13897 [cs.CV]. URL: https://arxiv.org/abs/2406.13897.

[39] Renrui Zhang et al. *Point-M2AE: Multi-scale Masked Autoencoders for Hierarchical Point Cloud Pre-training*. 2022. arXiv: 2205.14401 [cs.CV]. URL: https://arxiv.org/abs/2205.14401.

[40] Xiangdong Zhang, Shaofeng Zhang, and Junchi Yan. *PCP-MAE: Learning to Predict Centers for Point Masked Autoencoders*. 2024. arXiv: 2408.08753 [cs.CV]. URL: https://arxiv.org/abs/2408.08753.

[41] Qingnan Zhou and Alec Jacobson. *Thingi10K: A Dataset of 10,000 3D-Printing Models*. 2016. arXiv: 1605.04797 [cs.GR]. URL: https://arxiv.org/abs/1605.04797.