Challenges in 3D Data Synthesis for Training Neural Networks on Topological Features

Anonymous CVPR submission

Paper ID *****

Abstract

Topological Data Analysis (TDA) involves techniques of analyzing the underlying structure and connectivity of data. However, traditional methods like persistent homology can be computationally demanding, motivating the development of neural network-based estimators capable of reducing computational overhead and inference time. A key barrier to advancing these methods is the lack of labeled 3D data with class distributions and diversity tailored specifically for supervised learning in TDA tasks. To address this, we introduce a novel approach for systematically generating labeled 3D datasets using the Repulsive Surface algorithm, allowing control over topological invariants, such as hole count. The resulting dataset offers varied geometry with topological labeling, making it suitable for training and benchmarking neural network estimators. This paper uses a synthetic 3D dataset to train a genus estimator network, created using a 3D convolutional transformer architecture. An observed decrease in accuracy as deformations increase highlights the role of not just topological complexity, but also geometric complexity, when training generalized estimators. This dataset fills a gap in labeled 3D datasets and generation for training and evaluating models and techniques for TDA.

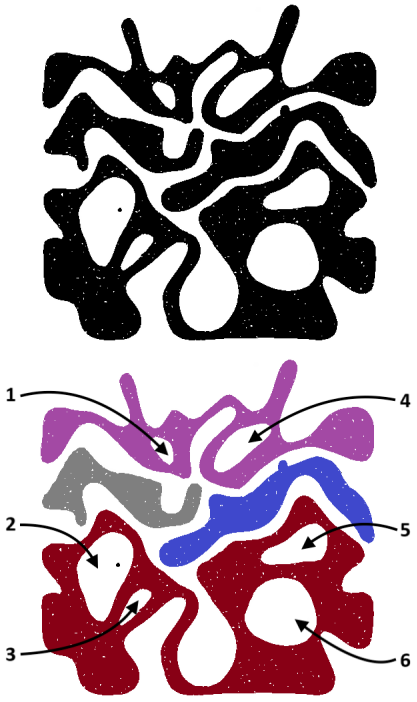

Figure 1. A 2D slice of a sample generated using the technique outlined in Section 3. The 2D binary image shows 6 holes across 4 disconnected objects Top: raw sample. Bottom: annotated analysis.

1. Introduction

The shape of an object is an essential feature that can be used for classification. Topology is the branch of mathematics that formalizes the study of shapes through a rigorous analysis of connectivity, cavities, and holes within an object. Topological Data Analysis (TDA), a relatively new field at the intersection of mathematics, computer science, and data science, focuses on analyzing the shape of data or objects [4, 7]. TDA offers powerful tools, such as Persistent Homology (PH), for understanding complex data by extracting topological features like n-dimensional holes, which uncover underlying patterns and relationships.

While Persistent Homology is a powerful tool, it has certain drawbacks, such as high computational cost, particularly for large datasets. To address this, machine learning techniques have been proposed and successfully applied to extract topological features and signatures [3, 6, 11, 12]. The underlying idea behind these studies is to bypass the persistent homology computation and directly estimate or predict topological invariants, such as *Betti numbers* and *genus*, through neural networks to reduce computational cost and gain additional insights. Additionally, neural networks have been employed in various other TDA tasks, including analyzing outputs from traditional techniques like

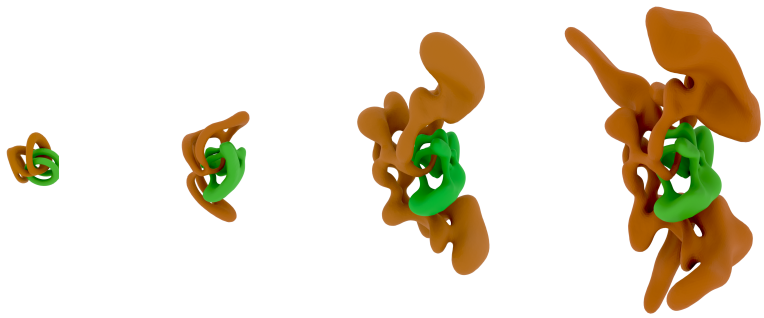

Figure 2. Random growth of interlinked genus 2 (green) and genus 3 (brown) objects using the method outlined in Sec. 3. Visualisation performed in Blender 3.0.1. [1]

persistent homology for applications such as medical diagnosis and classification [2, 13, 17].

Approaches that bypass PH computation and directly estimate topological invariants typically require large datasets for training and testing. The availability of such extensive data repositories is a major bottleneck in the further development of this area. Our current work addresses this issue by proposing a new method to synthetically generate complex and versatile labeled datasets suitable for the training and testing of neural networks in topological classification.

An example of a visual approach to estimating hole counts in 2D data can be seen in Figure, 1. This example is a 2D cross-section of the 3D generation process outlined in Section. 3.

Our method utilizes the Repulsive Surfaces algorithm [18] to perform homeomorphic (i.e., topology-preserving) deformations with randomized parameters and environmental constraints. This process generates a sequence of 3D data with known labels through an iterative growth mechanism, as illustrated in Figure 8. This data generation approach allows for:

- Incremental complexity in the generated data, making it suitable for both 3D or time-series 3D tasks. This also allows the accuracy of a model to be assessed at various geometrically complex stages while retaining the same topological complexities.
- Customizable growth configurations, allowing for adaptation to various applications and styles.

Our method fills a critical gap in accessible, topologically labeled 3D/4D data and has been utilized to train neural networks for TDA tasks using 'Betti Number' topological signatures (in a form we refer to as 'genus' in this paper, see Sec. 2). We demonstrate the efficacy of this synthetic data generation method through experiments with a 3D Convolutional Transformer Network (3DCTN) [9]. The dataset used in these experiments is called the Random Grid Repulse Dataset, or simply 'RG Repulse.' Details of the generation process are provided in Sec. 4.

1.1. Related Works

Previous studies have used a variety of datasets and output structures to perform TDA on raw input data. [11] used convolutional neural network architectures to estimate the betti numbers of 2D/3D data. The 2D model was trained on 2D image data consisting of randomly placed circles with randomized radii. For the 3D model, this was scaled into 3D with random spheres inside a 3D image. [6] scale up the idea of CNN-based Betti number estimation to 4D. A synthetic dataset was created that used cutouts of a solid 4D structure to introduce higher dimension holes. These cutouts were selected from predetermined objects that were randomly scaled and rotated. [15] introduces a deep learning model Pi-Net to estimate topological features directly from images. This paper used the datasets SVHN [10], CIFAR10, and CIFAR100 [8]. SVHN (Street View House Numbers) features photos of house numbers while CIFAR100 includes 100 common classes such as 'ship' or 'dog'. [3] introduces a deep learning model RipsNet for 3D point cloud TDA. This study sampled point clouds from the surface of the ModelNet10 dataset [16]. This is 3D mesh data which comprises of 10 classes such as 'Chair' or 'Bed'. [19] introduces a deep learning model TopologyNet to perform 3D point cloud TDA, similarly to RipsNet. This used the expanded ModelNet40 [16] which includes 40 classes instead of 10.

A key limitation across these studies is dataset availability. Datasets like CIFAR100, SVHN, and ModelNet40 contain strong correlations between the geometrical and topological properties of the objects. For example, recognizing that an object is a 'mug' inherently provides information about its toroidal topology. This correlation makes it difficult for networks to learn topological features independently of geometrical ones, potentially leading to overfitting. Additionally, these datasets possess fewer hole counts and limited topological complexity, restricting the variety of topological features. Finally, the distribution of samples in these datasets is optimized for object classification, not

for balancing topological features like Betti numbers.

[12] introduced the WFC Repulse dataset, which aimed to mitigate these limitations by generating synthetic data with varying topological complexity. This approach involved creating 3D scenes with multiple objects, each having a different genus (number of β_1 holes). This method allowed for controlled topological complexity, which was used to train transformer networks for TDA.

This paper builds upon the WFC Repulse dataset by expanding and refining the generation process. The differences include the selection of generation parameters, environment generation, post-processing, data format and hole counts. The previous method used uniform mesh-surface sampling in 4096 point clouds, while the proposed method converts the meshes to voxel cubes to better simulate use cases such as medical scans or material science scans. This voxel data can then be treated as a volumetric cube (3D voxel data) or converted to a point cloud for common neural network architecture pipelines.

2. Background in Topology

In this section we will briefly recall the basic notions in topology that are used in TDA and then describe the standard pipeline of Persistent Homology computation. For further reading see [4, 7].

A **geometric k-simplex** is the convex hull of $k+1$ affinely independent points in $\mathbf{R}^d$. For example, a point is a 0-simplex, an edge is a 1-simplex, a triangle is a 2-simplex, and a tetrahedron is a 3-simplex. A subset simplex is called a **face** of the original simplex. A **geometric simplicial complex** is a collection of geometric simplices that intersect only at their common faces and are closed under the *face* relation. See Figure 3 for an example. A **filtration** or a **filtered simplicial complex** is a sequence of nested simplicial complexes indexed by a scale parameter.

Homology, is a tool from algebraic topology to quantify the number of k-dimensional topological features (or holes) in a topological space, such as a simplicial complex. For instance, H_0 the zero degree homology classes describe the number of connected components, H_1 the one degree homology classes describe the number of loops, and H_2, the two dimensional homology classes quantifies voids or cavities. The ranks of these homology groups are referred to as **Betti numbers**. In particular, the Betti number β_k corresponds to the number of k-dimensional holes. In Figure 3 the Betti numbers are as follows, $\beta_0 = 1, \beta_1 = 1$, and $\forall k \geq 2, \beta_k = 0$. Table 1 shows the list of non-zero Betti numbers for all connected compact oriented 2-manifolds [14] along with other topological invariants such as *genus* g and *Euler Characteristic* $\chi = \beta_0 - \beta_1 + \beta_2$, which in this case are related as follows $g = \frac{\beta_1}{2} = -\frac{\chi-2}{2}$.

In a standard pipeline of *persistent homology* computation, a dataset, typically represented as a point cloud in a metric space, is used to first build a filtered simplicial complex which is used to construct a *boundary matrix* which is then finally reduced in a special form to read off the persistent homology. A common method for constructing such filtered simplicial complex is the *Vietoris-Rips* complex. This complex is formed by connecting data points that lie within a certain distance from each other, progressively increasing the complexity of the structure as the distance threshold grows. The idea of filtration is used to analyze data across multiple scales. As the filtration parameter (here the distance threshold) increases, new simplices are added, enabling the tracking of how topological features, such as Betti numbers, emerge and disappear across different scales.

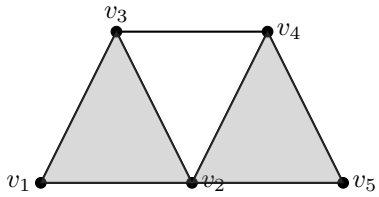

Figure 3. Example of a simplicial complex

Table 1. Genus (g), Betti numbers (β_n), and Euler characteristic (χ) of closed compact orientable surfaces.

Surface M	g	β_0	β_1	β_2	χ
Sphere S^2	0	1	0	1	2
Torus T^2	1	1	2	1	0
g-holed torus $T^2\#\ldots\#T^2$	g	1	$2g$	1	2-2g

3. Synthetic Data Generation

In a previous study, [12] created a synthetic dataset called the WFC Repulse dataset using the Wave Function Collapse algorithm [5] and Repulsive Surfaces algorithm [18]. This study demonstrated the viability of the proposed data generation method for topological training and assessed the segmentation ability of neural networks using topological signatures. Building on that, we describe how this method has been altered to produce a new RG Dataset.

We begin by outlining the basic steps of the data generation process. This process can be adapted to generate various types of datasets depending on the learning task.

1. *Seed*: This step manually creates a 'seed' object in the form of a 3D mesh with known topology.
2. *Environment*: Next, a randomly generated 'environment' was generated to grow the seed within. This environment

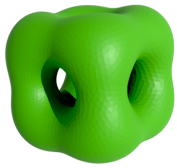

Figure 4. An example of a genus 5 seed.

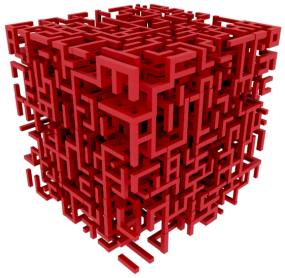

Figure 5. An environment generated using the random grid method.

Figure 6. A synthetic genus 5 object in mesh form with Voronoi surface displacement mapping.

will act as a constraint on which the seed will grow.

3. *Scene setup*: Then, the seed is randomly placed inside the environment using random placement and scaling.
4. *Deformations*: Next, the Repulsive Surfaces algorithm is used increase the seeds surface area within the environmental constraint. This process deforms the seed object without altering its topology.
5. *Subsampling*: Finally, the grown seed can be converted into a voxel or point cloud form with various noise/scaling applied.

The approach outlined above was common in both the previous study's WFC dataset [12] and the new RG Dataset. The key differences include the hole count which was raised from $[0-3]$ to $[0-20]$. The WFC Dataset was processed to a point cloud format via mesh surface sampling, while the RG Dataset was aimed to emulate 3D pixel data such as medical or material science scans. This involved voxelization and noise additions. The environment generation method was also changed from the wave function collapse algorithm to a new random grid approach. The random grid process is outlined below and the Wave Function Collapse algorithm can be seen in Appendix 6.2. The creation of a random grid allowed more flexibility and control in the scale, thickness and density of sections and used real number parameters for placement and thickness over discrete tile cells.

Now, we will explain each of the above outlined steps in more detail to generate the current RG Repulse dataset.

Seed We manually created 21 seed meshes for the RG Repulse dataset. Each of these meshes has a different number of 1-dimensional holes (β_1, genus), ranging from 0 to 20. See Figure 4 for an example genus 5 seed.

Random Grid Environment A unique environment is generated for every grown seed in the RG dataset. As the seed is grown within the environment, the geometry of the environment will determine aspects of the grown samples. We start this generation by dissecting a cube with a side length of 20 into 5^3 smaller cubic chunks, each with a side length of 4. Each of these smaller chunks is assigned different randomized parameters to create distinct sub-regions in the environment. Having different regions will produce geometric diversity as the sample in constrained in different ways. These random parameters include:

Axis resolution: A random resolution between 2 and 4 is selected for each axis in the subchunk, denoted as $\mathbf{x_{res}}$, $\mathbf{y_{res}}$, and $\mathbf{z_{res}}$. This produces a 3D grid of points comprising of $\mathbf{x_{res}} \times \mathbf{y_{res}} \times \mathbf{z_{res}}$ points for each subchuck separately.

Connection probability: For each subchunk we assign a probability P, randomly chosen between 0.15 and 0.25, referred to as the *edge connection probability*. Adjacent points are connected by edges with the probability P. This creates varied density in the overall environment as different subchuncks have different Ps.

Edge thickness: For each subchunk we assign a thickness T, chosen randomly between 0.4 and 0.6. Finally, we mesh the 3D environment by adding rectangular prisms around each edge, with thickness corresponding to the subchunks assigned value of T.

An example of the generated environment is shown in Figure 5.

Scene Setup Once the seed 3D mesh and the random 3D grid environment are ready, we place the seed meshes inside the environment. This is done by offsetting the bounding-box center of the seed to the environment's center. Then, the seed is *rotated* randomly between 0 and 2π around each axis. We perform an initial *global scaling* to adjust the surface area of the seed mesh to 1, followed by an additional random scaling of $\pm 25\%$. We then apply another set of anisotropic scaling of $\pm 50\%$ to each axis independently.

These augmented steps introduce variability to the initial seed conditions, allowing for further diversity in the grown samples.

Figure 7. Cross-sectional slices of a genus 5 object with Voronoi mesh displacement mapping and 3 octaves of 3D Perlin noise.

Deformations In this step, we aim to induce geometric complexity by forcing the surface area to increase within randomized constrained environments. To achieve this, we use the *Repulsive Surfaces (RS) algorithm* [18]. Intuitively, this is similar to growing a simpler 'seed' mesh into a larger, more complex form through iterative *homeomorphic* deformations. Typically, the RS algorithm is used to deform a complex manifold into a simpler equivalent form; however, in our approach, we reverse this goal and use it to add more complexity. The algorithm works by pushing apart pairs of points in an attempt to maintain a uniform distribution. This is combined with energy minimization, which is determined by the relationship between a points spacial distance and surface distance (curved surfaces have shorter pathways in spatial coordinates than surface coordinates), which penalizes objects with greater variance, see [18].

Since the generated sample is in mesh form, many surface mapping deformations can be applied. We used Blender [1] to apply a Voronoi displacement map, a method that subdivides a surface into regions based on proximity to a set of points, resulting in a distinct, organic-looking pattern. Voronoi is particularly useful for creating unique surface geometries, adding variation and breaking up the smoothness of the mesh. This displacement was applied with an intensity of 0.5 and a size of 0.1 to introduce surface variations, centered around the midlevel of the mesh. This approach was chosen to ensure each mesh has a unique, textured appearance, though other texture maps or real-world images could be used depending on the desired application.

Subsampling The final mesh, after all deformations, was sampled into a 3D voxel space with a 256^3 resolution. We then generated three octaves of Perlin noise, a gradient noise algorithm commonly used in procedural texture generation, and applied them to the voxel space. Perlin noise creates smooth, continuous variations, which are particularly effective for adding natural-looking randomness to the voxel grid. Each octave represents a layer of noise with different frequencies and intensities, allowing for more complex surface deformations. In this context, the *scale* adjusts the size of the noise patterns, with lower values producing larger, more prominent features. The *threshold* defines which parts of the noise should be considered significant enough to alter the voxel space. The first octave using a scale of 4 and threshold of 0.5 which was added to the cube. The second octave using a scale of 8 and threshold of 0.55 which was added to the cube. The third octave using a scale of 16 and threshold of 0.55 which was subtracted from the cube. Afterwards, a *smoothing* step was performed by applying a Gaussian filter to the 3D voxel data with a standard deviation $\sigma = 0.25$ to achieve a smoother representation. These noise, scale, and resolution settings are easily configurable for different objectives. An example result is illustrated in Figure 7. These 2D slices are cross-sections of the sample shown in Figure 6. The generation time of these samples can be seen in Appendix 6.3. Next, we provide the specifications for the final generated RG Repulse data.

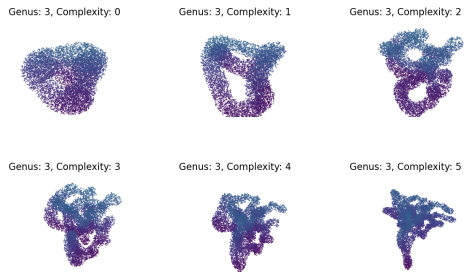

Figure 8. Examples of complexity levels for a genus 3 sample in the RG dataset. Each sample contains 8192 points. Complexity levels $[0-5]$ for objects of genus 10 can also be seen in Appendix 6.1

RG Dataset Specifications The 21 seed meshes, each consisting of 1-dimensional holes ranging from 0 to 20, were grown in unique random environments with surface area increases ranging from 15–20x. Complexity levels between 0 and 5 were assigned at each 20% of the samples overall growth. See Figure 8 for example complexity levels. These samples were then converted into voxel cubes with a 256^3 resolution. Finally, the RG dataset comprises 6,366 training samples and 1,456 test samples.

Table 2. Class (hole count) vs complexity accuracy (%) of 3DCTN (soft heatmap version).

Class	0	1	2	3	4	5
0	100.00%	93.33%	86.67%	66.67%	66.67%	60.00%
1	100.00%	80.00%	60.00%	40.00%	40.00%	20.00%
2	87.50%	68.75%	62.50%	50.00%	56.25%	43.75%
3	100.00%	60.00%	30.00%	10.00%	10.00%	10.00%
4	100.00%	92.31%	69.23%	46.15%	30.77%	38.46%
5	100.00%	94.44%	55.56%	44.44%	38.89%	44.44%
6	100.00%	81.82%	63.64%	54.55%	54.55%	54.55%
7	92.31%	84.62%	53.85%	30.77%	23.08%	23.08%
8	100.00%	100.00%	75.00%	75.00%	50.00%	75.00%
9	66.67%	66.67%	33.33%	33.33%	33.33%	16.67%
10	76.92%	84.62%	84.62%	69.23%	53.85%	46.15%
11	100.00%	100.00%	53.85%	38.46%	23.08%	15.38%
12	100.00%	100.00%	84.62%	69.23%	61.54%	46.15%
13	100.00%	100.00%	100.00%	75.00%	75.00%	58.33%
14	100.00%	100.00%	100.00%	36.36%	45.45%	36.36%
15	100.00%	100.00%	100.00%	100.00%	90.91%	72.73%
16	100.00%	100.00%	80.00%	60.00%	60.00%	50.00%
17	100.00%	100.00%	100.00%	78.57%	71.43%	71.43%
18	90.00%	100.00%	90.00%	60.00%	50.00%	30.00%
19	100.00%	100.00%	100.00%	90.91%	90.91%	81.82%
20	58.33%	66.67%	58.33%	58.33%	41.67%	41.67%

4. Experimental Results

We used our labeled dataset 'RG Repulse,' generated using the algorithm described in the previous section, to train a 3D Convolutional Transformer Network (3DCTN, [9]) for topological classification based on Betti numbers, specifically β_1. This section outlines the experimental results regarding the model training and performance. The parameters used for training are detailed in Table 3.

Results The 256^3 resolution voxel cubes in the RG dataset were uniformly sampled into 8,192 points to generate a sparser point cloud. This subsampling requires fewer computational resources than the original 16 million points. The 3DCTN results on the test set are shown in Table 2.

Table 3. Training parameters

Parameter	Value	Parameter	Value
Model	3DCTN	Learning rate	0.01
Optimizer	SGD	Weight decay	0.0001
Point count	8192	Epoch	300

5. Discussion and Conclusion

The results of the RG experiment demonstrated a decrease in accuracy as complexity levels increased. This suggests that greater homeomorphic deformation introduces more variability and challenge within the samples. Such variability is ideal for training neural networks for TDA tasks, which are invariant to geometric differences. The diversity in appearance across topologically equivalent samples can help mitigate overfitting. Additionally, this dataset has potential utility beyond machine learning models, including for the evaluation of persistent homology algorithms.

Previous experiments [12] conducted on the WFC dataset—which featured samples with lower hole counts, smoother surfaces, and point cloud formats—demonstrated the capability of transformers to segment data using topological labels. Per-point segmentation enables each object within a scene to be identified, classified, and localized. This is valuable because it allows for the preservation and analysis of relationships between metric, geometric, and topological properties such as size, volume, and shape.

By adjusting parameters like seeds, growth rates, sample sizes, hole sizes, and post-processing techniques, we gain substantial control over the appearance and properties of the generated samples. This flexibility is evident in the differences between the WFC and RG datasets. For real-world applications, this process can be tailored to replicate key

aspects of actual data. Consequently, transfer learning can then be applied using smaller real-world datasets where after some understanding of topological structures exists.

Topological Data Analysis (TDA) is a rapidly growing field with an increasing need for labeled datasets. The data generation method presented in this paper aims to address this gap by providing data that is both rich in topological variety and appropriately labeled. Our experiments demonstrate the viability of this dataset generation technique, as well as the inherent challenges associated with it. Conceptually, TDA presents different challenges compared to conventional classification tasks due to the significant visual differences between objects of the same topological class. This synthetic data allows researchers to explore and evaluate specific topological features without interference from extraneous variables that may be present in real-world data.

References

[1] Blender Development Team Blender Foundation. Blender 3.0.1, 2021. Software. 2, 5

[2] Yu-Min Chung, Chuan-Shen Hu, Yu-Lun Lo, and Hau-Tieng Wu. A persistent homology approach to heart rate variability analysis with an application to sleep-wake classification. *Frontiers in Physiology*, 12:637684, 2021. 2

[3] Thibault de Surrel, Felix Hensel, Mathieu Carrière, Théo Lacombe, Yuichi Ike, Hiroaki Kurihara, Marc Glisse, and Frédéric Chazal. Ripsnet: a general architecture for fast and robust estimation of the persistent homology of point clouds. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pages 96–106. PMLR, 2022. 1, 2

[4] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010. 1, 3

[5] Maxim Gumin. Wavefunctioncollapse. *GitHub repository*, 2016. 3, 8

[6] Khalil M. Hannouch and Stephan Chalup. Learning to see topological properties in 4d using convolutional neural networks. In *Proceedings of 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML), PMLR*, pages 437–454, 2023. 1, 2

[7] Allen Hatcher. *Algebraic Topology*. Cambridge University Press, Cambridge, UK, 2002. 1, 3

[8] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009. 2

[9] Dening Lu, Qian Xie, Kyle Gao, Linlin Xu, and Jonathan Li. 3dctn: 3d convolution-transformer network for point cloud classification. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2022. 2, 6

[10] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature iearning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. 2

[11] Rahul Paul and Stephan Chalup. Estimating betti numbers using deep learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019. 1, 2

[12] Dylan Peek, Matt Skerritt, and Stephan Chalup. Synthetic data generation and deep learning for the topological analysis of 3d data. In *2023 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2023. In Press, accepted September 2023. 1, 3, 4, 6

[13] Matteo Rucco, Lorenzo Falsetti, Damir Herman, Tanya Petrossian, Emanuela Merelli, Cinzia Nitti, and Aldo Salvi. Using topological data analysis for diagnosis pulmonary embolism. *arXiv preprint arXiv:1409.5020*, 2014. 2

[14] Herbert Seifert and William Threlfall. *Lehrbuch der Topologie*. Teubner, Leipzig, 1934. 3

[15] Anirudh Som, Hongjun Choi, Karthikeyan Natesan Ramamurthy, Matthew P Buman, and Pavan Turaga. Pi-net: A deep learning approach to extract topological persistence images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 834–835, 2020. 2

[16] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 2

[17] Takehiko Yamanashi, Mari Kajitani, Masaaki Iwata, Kaitlyn J Crutchley, Pedro Marra, Johnny R Malicoat, Jessica C Williams, Lydia R Leyden, Hailey Long, Duachee Lo, et al. Topological data analysis (tda) enhances bispectral eeg (bseeg) algorithm for detection of delirium. *Scientific Reports*, 11(1):1–9, 2021. 2

[18] Chris Yu, Caleb Brakensiek, Henrik Schumacher, and Keenan Crane. Repulsive surfaces. *arXiv preprint arXiv:2107.01664*, 2021. 2, 3, 5

[19] Chi Zhou, Zhetong Dong, and Hongwei Lin. Learning persistent homology of 3d point clouds. *Computers & Graphics*, 102:269–279, 2022. 2

6. Appendix

6.1. Complexity levels

This section includes figures for objects of genus 3 and genus 10 and their corresponding complexity levels within the RG dataset.

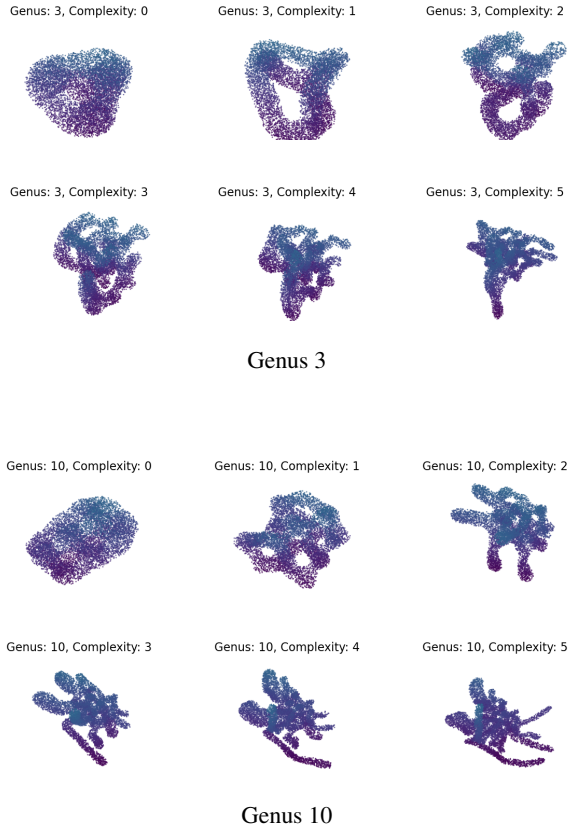

Figure 9. Examples of complexity levels 0–5 for genus 3 and 10 in the RG dataset. Each sample contains 8192 points.

6.2. Wave Function Collapse Algorithm

The following is psuedo code for a tile based implementation of the WFC algorithm. It involves the predetermination of tiles and subsequent adjacency rules.

Initialize grid with uncollapsed cells;
Initialize tile set with all possible tiles and their neighbour rules;
while *there are uncollapsed cells in the grid* **do**
  Select the cell with the lowest entropy (least number of possible tiles);
  if *there are multiple cells with the same entropy* **then**
    Select one randomly;
  end
  Collapse the selected cell by choosing a tile randomly from its possible tiles;
  `// Propagate constraints`
  for *each neighbour of the collapsed cell* **do**
    Update the neighbour's possible tiles based on the neighbour rules;
    if *the neighbour's possible tiles list changes* **then**
      Mark the neighbour for further constraint propagation;
    end
  end
  Propagate constraints recursively until no further changes occur;
end
if *grid is fully collapsed* **then**
  return completed grid;
end
else
  handle contradiction (e.g., restart or backtrack);
end

Algorithm 1: Wave Function Collapse Algorithm (Tile-Based) [5]

6.3. RG Dataset Generation Time

Table 4. Average computation time for different genus manifolds in the 'RG Repulse' dataset (minutes) using 24-core Intel Xeon Scalable 'Cascade Lake' processors. Samples were grown across multiple CPUs in parallel with each sequence being allocated 2 cores.

Genus 0-6	g_0	g_1	g_2	g_3	g_4	g_5	g_6
Average Time (min)	117.6	51.3	76.4	96.2	139.5	150.3	175.3
Genus 7-13	g_7	g_8	g_9	g_{10}	g_{11}	g_{12}	g_{13}
Average Time (min)	192.1	208.1	209.2	211.1	226.8	232.1	227.9
Genus 14-20	g_{14}	g_{15}	g_{16}	g_{17}	g_{18}	g_{19}	g_{20}
Average Time (min)	237.6	235.7	233.5	221.9	222.8	220.6	223.83