Learning to See Topological Properties in 4D Using Convolutional Neural Networks

Khalil Mathieu Hannouch [*1] **Stephan Chalup** [*1]

Abstract

Topology describes the essential structure of a space, and in 4D, a larger variety of topologically distinct manifolds can be embedded versus 2D or 3D. The present study investigates an end-to-end visual approach, which couples data generation software and convolutional neural networks (CNNs) to estimate the topology of 4D data. A synthetic 4D training data set is generated with the use of several manifolds, and then labelled with their associated Betti numbers by using techniques from algebraic topology. Several approaches to implementing a 4D convolution layer are compared. Experiments demonstrate that already a basic CNN can be trained to provide estimates for the Betti numbers associated with the number of one-, two-, and three-dimensional holes in the data. Some of the intricacies of topological data analysis in the 4D setting are also put on view, including aspects of persistent homology.

1. Introduction

Convolutional neural networks (CNNs) share common features with the human visual cortex (Nonaka et al., 2020; Xu & Vaziri-Pashkam, 2021) and have shown superior performance when compared to humans in many computer vision tasks (Krizhevsky et al., 2012). The aim of the present project is to demonstrate that CNNs can be trained to 'see' the topology of basic simulated data in four dimensions (4D), similarly as humans can visually analyse the shape and topology of basic objects in 3D.

4D data can occur, for example, if 2D or 3D data is equipped with other dimensions. It can also be the outcome of manifold learning, where high-dimensional data is reduced to a low-dimensional representation that still contains all essential information (Lee, 2000). According to the Poincaré-Hopf theorem (Brasselet et al., 2009), there is an important relationship between the global topology of manifolds and the characteristics of dynamical systems on them. However, the number of studies where non-trivial topological features occur in 4D data is still low (Aziz et al., 2019; Carlsson et al., 2008; Reimann et al., 2017). This is probably due to a lack of labelled manifold data, and the limitations of computational approaches used to analyse them.

For humans, one approach to visualising 3-dimensional manifolds (3-manifolds), such as spheres and various tori, in 4D is by slicing them into a sequence of 3D sections. Figure 1 is a MATLAB (MATLAB, 2017) visualisation of $S^1 \times S^1 \times S^1$ in an (x, y, z, w)-system. A formula to generate this 3-manifold is provided in Section 4.2, and additional visualisations are included in Appendix A.

A better understanding and new computational tools to address the rich topological structure of manifolds in 4D and their associated dynamical systems has the potential to offer new insights in physics and science, see for example (Hofmann et al., 2018; Scott et al., 2019; Sagristà et al., 2017).

2. Approach and potential applications

In this study, we consider 4D cubes (4D images), and use a selection of fundamental 3-manifolds, such as spheres and tori of potentially different topologies and shapes, to design and introduce cavities into the cube. A 4D CNN is then trained to estimate how many holes of different dimensions occur in the cube. The task could be caricatured as counting the various types of tunnels, bubbles, and cavities in a 4D block of Swiss cheese. In algebraic topology, these essential features of the space can be captured by the concept of homology, and more specifically, the Betti numbers (Edelsbrunner & Harer, 2010).

The intuition behind employing cubes with cavities follows from data that arise in disciplines such as radiology and material science, which can be construed as higher-dimensional analogues of squares (2D images). Real-time ultrasound

[*]Equal contribution [1]School of Information and Physical Sciences, University of Newcastle, Callaghan NSW 2308, Australia. Correspondence to: Khalil Mathieu Hannouch <khalil.hannouch@uon.edu.au>, Stephan Chalup <stephan.chalup@newcastle.edu.au>.

Proceedings of the 2nd Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

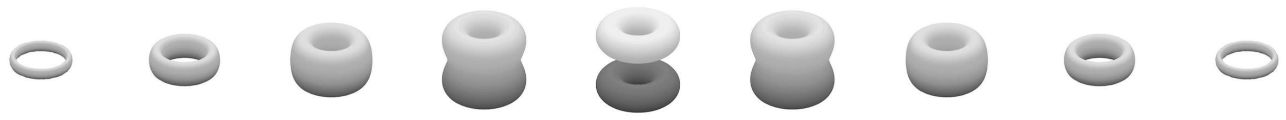

Figure 1. For visualising $S^1 \times S^1 \times S^1$, 3D slices were taken along the w-axis, and ordered from left to right, with the central image depicting the slice at $w = 0$ (see also Appendix A).

can be considered as 3D data, just as fMRI is an example of how a 3D target can be captured over time to produce 4D data. These images are often dense, in the sense that they capture large regions of target material, in comparison to empty space. For example, when radiography is used for medical diagnosis, radiographic films generally detect hard and soft tissue; regions that comprise predominately of fluid- or air-filled cavities, such as within bladders and sinuses, could then be taken as features of an image. In material science, micro-tomography has been used in order to observe the 4D structural evolution of cement paste in situ during a hydration process (Zhang et al., 2022). Similarly, the properties of materials that possess cavities by design, such as metallic and syntactic foams (Al-Sahlani et al., 2018; Duarte et al., 2020), can be studied by observing how their structure (topology) changes over time when exposed to a mechanical force, temperature, or electrical current, or conversely, how manufacturing processes affect their structure (Kassab et al., 2022; Saadatfar et al., 2017; Ando et al., 2021). An analysis of the topological characteristics of medical imaging is also a research consideration (Loughrey et al., 2021; Kim et al., 2019; Stolz et al., 2021).

Thus, we make a connection between computer vision, and how it can be used to visually explore some of the topological aspects of materials as captured in data in the form of higher dimensional images. It is in using these denser formats that existing methods begin to exhibit complexity issues in practice, and as we discuss in Section 5, this is distinct from exploring the topology of data that are presented in the form of point clouds, or meshes and triangulations that are constructed over point clouds. Practically, we propose that once a dataset, which sufficiently models real data (Gao et al., 2022), is generated, it may be possible to train a CNN that is capable of estimating the Betti numbers of the data. In order to focus our investigation and provide an introduction to our approach, we restrict ourselves to simulated, single-component, samples that are well-understood (Section 4.3), and we implement only simple CNN arrangements (Section 6). The latter choice highlights the analogy of our approach to 4D vision, and is supported by the relatively high brain hierarchy score of simple CNNs (Nonaka et al., 2020).

3. Related work

Among the studies that combine machine learning and persistent homology concepts are Som et al. (2020) and Zhou et al. (2021), who propose the use of a functional approximation of persistence diagrams, called persistence images (Adams et al., 2017), and demonstrate how deep learning can be applied to the task of inferring persistence images. These works run parallel to the efforts of de Surrel et al. (2022), who demonstrate the use of a network, called RipsNet, to predict the persistence images and landscapes associated with 2D and 3D data; some basic synthetic datasets are also considered.

The approach and data representation of our study was inspired by the work of Paul & Chalup (2019), who used 2D and 3D simulated data cubes from which basic manifolds were cut out. Standard 2D and 3D CNNs were trained to predict the Betti numbers of this data, while the persistent homology software javaPlex (Adams et al., 2014) experienced complexity issues with increasing data size.

In the 4D data setting, CNNs have been applied to spatio-temporal data (three spatial and one time dimension) (You & Jiang, 2018; Zhang et al., 2020), and 4D fMRI time series data (Noor et al., 2020). 4D CNN models have also been used for CT image reconstruction (Clark & Badea, 2019) and segmentation (Myronenko et al., 2019). You & Jiang (2018) worked on human action recognition using calibrated RGB-D data, which were then converted into sequences of 'solid' 3D representations. The models themselves were based on 3D CNNs, and used recurrent neural networks to aggregate temporal information; a true 4D CNN approach was not used. In contrast, Zhang et al. (2020) used a 4D convolution component in their networks, which captured the dependencies between short clips that were sampled from a video. The authors explain that the computing and memory demands of 4D CNNs can be relatively high, and chose to use $k \times k \times 1 \times 1$ and $k \times 1 \times 1 \times 1$ kernels only, in order to reduce both the number of training parameters, and the risk of overfitting. The choice could also be seen to be reasonable, given the true (video) nature of their data. In our study, we intend to employ true 4D kernels in which all dimensions are greater than 1. Gessert et al. (2020) considered a more efficient variant of ResNet4D, called facResNet4D, that used factorisation to decompose the kernel into separate spatial and temporal kernels. The authors explain that this modification reduces the number of parameters, but also reduces the representative power of the model. Choy et al. (2019) generalised the sparse convolution (Graham, 2014; Graham & van der Maaten, 2017) and proposed a 4D application, called the Minkowski CNN. A sparse tensor representation was used to encode only viable points within a sample, and each of these points was attributed a feature vector. Impressive results were demonstrated on several spatio-temporal datasets, however, these consisted mostly of scans of indoor and outdoor spaces, which are often captured densely, but are inherently sparse.

It appears that 4D deep learning is still in its infancy, and restricted to a small number of studies with specialised CNNs that were designed or optimised for specific sparse or temporal data applications. The present study proposes a CNN (Section 6) that can be regarded as a 4D generalisation from standard 2D or 3D CNNs. We also aimed at using topologically more complex data than previous studies.

4. Data generation

A key component of our CNN-based computer vision approach towards topology recognition was the generation of labelled 4D training data. Our data comprised of 4D cubes, into which we introduced cavities by cutting out 4D balls, that is, $B^n = \{x \in \mathbb{R}^n; ||x|| \leq 1\}$ for $n = 4$, and various versions of tori that exist in 4D, including $S^1 \times B^3$, $S^2 \times B^2$, and $S^1 \times S^1 \times B^2$; we did not consider their connected sums. All four dimensions of these data were treated equally, and one could indeed inspect these data from any 4D perspective, without necessarily assuming that they arise from observing 3D phenomena evolving with time.

4.1. Homological results to obtain ground truth labels

Topology describes the essential properties of a space that remain invariant under homeomorphic deformations. Manifolds can be regarded as the central topological spaces that are investigated in much of topology and geometry (Lee, 2000). The classification of 2D compact manifolds is a classical result (Gallier & Xu, 2013), and says that all compact 2D manifolds are homeomorphic to the sphere S^2, a connected sum of tori, or a connected sum of projective spaces. If only orientable surfaces are considered, then there exist up to homeomorphism only the sphere and genus g surfaces (g-holed tori). A corresponding result for classifying 3-manifolds is much more involved (Bessieres et al., 2010).

The Betti numbers capture the essential structure given by the holes of a manifold or topological space (Edelsbrunner & Harer, 2010), and are a more fine-grained signature than the more broadly known Euler characteristic χ. The k^{th} Betti number is often denoted with β_k. In $\mathbb{R}^4$, only the first four Betti numbers are relevant, and the relationship to the Euler characteristic is given by $\chi = \beta_0 - \beta_1 + \beta_2 - \beta_3$, where β_0 is the number of path-connected components, β_1 is the number of 1D (circular) holes, or independent tunnels, β_2 is the number of 2D holes, cavities, or bubbles, and β_3 is the number of 3D holes.

Holes themselves are not technically part of a topological space, however, they are an invariant feature. A crude way to understand the dimension of a hole is to consider the dimension of the manifold that we find at the ‘boundary’ of a hole. For example, suppose that our topological space is a 2D plane I^2. By removing the interior of a disc $\text{Int}B^2$ from the interior of the plane, we find a circular boundary S^1, which is a 1-manifold. In this case, we have introduced a 1D hole, and β_1 is incremented by 1. This intuition can be extended to k-dimensional holes, and is formalised by homology, where roughly speaking, a k-dimensional *cycle* is a closed submanifold, a k-dimensional *boundary* is a cycle that is also the boundary of a submanifold, and a k-dimensional *homology class* is an equivalence class of the group of cycles modulo the group of boundaries Z_k/B_k, otherwise known as the k^{th} homology group H_k. Any non-trivial homology class represents a cycle that is not a boundary, or equivalently, a k-dimensional hole. The k^{th} Betti number β_k can be defined as the rank of H_k (Edelsbrunner & Harer, 2010).

Table 1. Betti numbers β_i and the Euler characteristic χ of selected low-dimensional manifolds. The manifolds involving a subtraction from I^4 were the most relevant to this work.

Manifold	β_0	β_1	β_2	β_3	χ
Circle S^1	1	1	0	0	0
2-Ball B^2	1	0	0	0	1
2-Sphere S^2	1	0	1	0	2
Torus $T^2 = S^1 \times S^1$	1	2	1	0	0
$S^1 \times B^2$	1	1	0	0	0
3-Ball B^3	1	0	0	0	1
3-Sphere S^3	1	0	0	1	0
$S^1 \times S^2$	1	1	1	1	0
$S^1 \times S^1 \times S^1$	1	3	3	1	0
4-Ball B^4	1	0	0	0	1
$I^4 - B^4$	1	0	0	1	0
$S^1 \times B^3$	1	1	0	0	0
$I^4 - (S^1 \times B^3)$	1	0	1	1	1
$S^2 \times B^2$	1	0	1	0	2
$I^4 - (S^2 \times B^2)$	1	1	0	1	-1
$S^1 \times S^1 \times B^2$	1	2	1	0	0
$I^4 - (S^1 \times S^1 \times B^2)$	1	1	2	1	1

Some fundamental manifolds and their Betti numbers are listed in Table 1. In order to calculate these Betti numbers, we appealed to singular homology, and applied both the Künneth theorem, which relates the homology groups of two topological spaces to those of their product space, and the Mayer-Vietoris Sequence, which relates the homology groups of a space to those of its subspaces (Hatcher, 2002); these theorems provide a means to handle operations corresponding to taking products and differences, respectively. β_0 is often interpreted as the number of connected components, however, it would seem more consistent with the higher dimensional Betti numbers if $\beta_0 = 1$ implied the existence of a 'gap' between two components. Addressing this is the rationale behind *reduced* homology, and its application can simplify some calculations. For this reason, we employed the reduced Mayer-Vietoris Sequence in order to find the isomorphism necessary to calculate the 0th homology groups, and deduce their Betti numbers. More detail about the Künneth theorem, the Mayer-Vietoris Sequence, and reduced homology are provided in Appendix C.

We demonstrate an application of these ideas by calculating the Betti numbers of $I^4 - (S^1 \times S^1 \times B^2)$. Since the homology groups of the factors of $S^1 \times S^1 \times B^2$, and its boundary $S^1 \times S^1 \times S^1$, are well-known, it can be shown that the Tor functor components in the Künneth theorem's short exact sequences are trivial, which implies the following isomorphisms

$$H_n(S^1 \times S^1 \times B^2) \cong \begin{cases} \mathbb{Z} & n = 0, 2 \\ \mathbb{Z}^2 & n = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$H_n(S^1 \times S^1 \times S^1) \cong \begin{cases} \mathbb{Z} & n = 0, 3 \\ \mathbb{Z}^3 & n = 1, 2 \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

We then define an embedding $\varphi : S^1 \times S^1 \times B^2 \to I^4$, and let $K = \varphi(S^1 \times S^1 \times B^2)$ and $X = I^4 - K$. We also define $Y = K \cup N(K)$, where $N(K)$ is an open neighbourhood of K, so that we have $X \cup Y = I^4$, and the homotopy equivalence $X \cap Y \simeq S^1 \times S^1 \times S^1$. The Mayer-Vietoris Sequence is then applied using X and Y in light of the isomorphism between the homology groups of homotopy equivalent spaces in singular homology (Hatcher, 2002, Chapter 2). For the $n = 0$ case, we apply the reduced Mayer-Vietoris sequence, and collectively, these results imply that

$$H_n(I^4 - S^1 \times S^1 \times B^2) \cong \begin{cases} \mathbb{Z} & n = 0, 1, 3 \\ \mathbb{Z}^2 & n = 2 \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Thus, $\beta_n = 1$ for $n = 0, 1, 3$, $\beta_2 = 2$, and all other Betti numbers are 0.

The Betti numbers for the other examples in Table 1 were obtained in a similar manner, and were aggregated to produce a label for the samples from which several of these objects were cut out.

4.2. Implicit representation of topological objects

In order to describe the 4D objects of interest, we began by deriving parametric equations to describe the 3-manifolds that constituted their boundaries (Table 2). Note that while $S^1 \times B^3$ and $S^2 \times B^2$ have homeomorphic boundaries, we required distinct representations, and use the convention that $S^1 \times S^2$ is the boundary of $S^1 \times B^3$, and $S^2 \times S^1$ is the boundary of $S^2 \times B^2$. The parametric equation for the 3-sphere S^3 is well known. The remaining equations were derived trigonometrically, by rotating the second factor around the first factor. For example, the parametric equation for $S^1 \times S^2$ was derived by rotating S^2 around S^1. For $S^1 \times S^1 \times S^1$, the torus $S^1 \times S^1$ was rotated around S^1. Figure 1 is a visualisation of $S^1 \times S^1 \times S^1$ with $\alpha = 0$.

These equations were subsequently compressed into implicit formulas (see Appendix D for an example). Then, by replacing the equality in each formula with an inequality, as shown in Table 2, we were able to describe a 'solid' object that could be removed from the interior of a 4D cube.

4.3. Data generation software

Data generation software was implemented in Python using data structures from the NumPy library (Harris et al., 2020). Each sample began as a 4D tensor with each dimension having an equal length I, and every entry set to 1; this represented an I^4 toxel image, where 'toxel' refers to the 4D analogue of a pixel, and may also be referred to as a 4D-voxel or hyper-pixel in literature. A random number of cavities were then introduced into a sample by setting the toxels that represented the cavities to 0. This produced what could then be seen as a 4D generalisation of a black-and-white image. Each cavity took a form that was homeomorphic to one of the objects in Table 2; the objects were randomly scaled, oriented, and positioned, using standard rotation, scaling, and translation operations. Since the Betti numbers of each cavity within a sample were known, a label could be produced by simply summing these Betti numbers over their dimensions; using persistent homology software was not necessary.

We restricted our experiments to samples comprising of only one connected 4D toxel cube component ($\beta_0 = 1$), and did not allow tunnels to traverse to and from the cube's boundary, or allow divots to form at the boundaries; a cube's boundaries were never cut away. Furthermore, cavities were

Table 2. Parametric equations in an (x, y, z, w)-system, and the implicit formulas that were used to produce 4D cavities from the manifolds that are listed in row one.

S^3	$S^1 \times S^2$	$S^2 \times S^1$	$S^1 \times S^1 \times S^1$
$\begin{pmatrix} a\sin\gamma\sin\theta\sin\phi \\ a\sin\gamma\sin\theta\cos\phi \\ a\sin\gamma\cos\theta \\ a\cos\gamma \end{pmatrix}$	$\begin{pmatrix} (a\cos\theta + r)\cos\gamma \\ (a\cos\theta + r)\sin\gamma \\ a\sin\theta\cos\phi \\ a\sin\theta\sin\phi \end{pmatrix}$	$\begin{pmatrix} (r + a\cos\gamma)\sin\theta\cos\phi \\ (r + a\cos\gamma)\sin\theta\sin\phi \\ (r + a\cos\gamma)\cos\theta \\ a\sin\gamma \end{pmatrix}$	$\begin{pmatrix} (R - B(r + a\cos\theta)\cos\phi + Aa\sin\theta)\cos\gamma \\ (R - B(r + a\cos\theta)\cos\phi + Aa\sin\theta)\sin\gamma \\ (r + a\cos\theta)\sin\phi \\ A(r + a\cos\theta)\cos\phi + Ba\sin\theta \end{pmatrix}$
$x^2 + y^2 + z^2 + w^2 \leq a^2$	$(\sqrt{x^2 + y^2} - r)^2 + z^2 + w^2 \leq a^2$	$(\sqrt{x^2 + y^2 + z^2} - r)^2 + w^2 \leq a^2$	$(\sqrt{(-B(\sqrt{x^2+y^2} - R) + Aw)^2 + z^2} - r)^2 +$ $(A(\sqrt{x^2+y^2} - R) + Bw)^2 \leq a^2$, where $A = \cos\alpha$ and $B = \sin\alpha$

not allowed to intersect each other. In order to observe these rules, a 1 toxel boundary and minimum 5 unit spacing between cavities was implemented.

The design of the 4D dataset presented in this work was the result of several iterations of variations. The cavity parameters, namely, the minimum radii, were guided by the observations discussed in Section 5 that allowed the persistent homology software to run error free. Initially, a 16^4 cube was considered, however it was discovered to be too small to fit the more expansive variations of $S^1 \times S^1 \times B^2$, say when $\alpha = \pi/2$. A 32^4 cube was large enough to fit several instances of these objects, but small enough for our network and hardware to cope with (see Sections 6 and 7). An example is shown in Appendix B.

For humans, the skill of counting is straightforward in 2D and even 3D. Many techniques may be employed to count or track items. In smaller cases, we may even rely on the psychological phenomenon, called *subitizing*, whereby one can quickly, and accurately judge the number of items, rather than explicitly counting them (Kaufman et al., 1949). For example, it may be possible to subitize a random arrangement of 1 to 6 dots. In larger examples, counting can become challenging and time consuming, and one cannot generally rely on subitizing in cases containing much more than 4 to 6 objects (Kaufman et al., 1949). Unfortunately, data are not generally limited to such small problems, therefore, in order to test the utility of CNNs, the introduction of up to 16 holes into a sample was considered to be sufficiently challenging.

The parameters of the 4D dataset that we finally selected to demonstrate our approach are summarised in Table 3. Note that for $S^1 \times S^1 \times B^2$, the parameter α is used to find A and B, as defined in Section 4.2. This dataset was generated in parallel on a High Performance Computing (HPC) Grid in 1000-sample batches, over 32 nodes. An average of 2.43 hours was required for each batch, and the entire process utilised approximately 77.69 HPC hours.

Additional earlier experiments, which employed less diverse datasets, did demonstrate that CNNs were capable of estimating many more than 16 holes. A 3D CNN with a very similar design to the 4D CNN described in Section 7 achieved >99% accuracy when performing inference on samples with up to 16 2D holes alone, and up to 32 holes of mixed dimension, which were introduced by cutting out balls B^3 and $S^1 \times B^2$. Pilot experiments that employed our 4D CNN also demonstrated that it was capable of estimating β_3 up to 16, with approximately 97% accuracy, when tested on a dataset with only ball B^4 cavities. Therefore, it is certainly possible that the potential of 4D CNNs is far greater. The 16 hole restriction that we imposed was a consequence of introducing more exotic, and expansive, cavities into a 32^4 toxel space, while taking the computational restrictions of our hardware into account.

5. Persistent homology

Persistent homology is a computational approach to computing homology, and forms part of a larger tool set, known as Topological Data Analysis (Edelsbrunner & Harer, 2010; Otter et al., 2017). It can be applied in cases where the dataset is a point cloud that is sampled from an underlying manifold, and can create a filtration of simplicial complexes from which an estimate of topological indices, such as the Betti numbers of the underlying manifold, can be derived. The computational complexity of the simplicial approximations grows significantly with the number of data points, and hence scaling in applications in higher dimensions becomes an issue (Edelsbrunner & Harer, 2010; Paul & Chalup, 2019; Zhou et al., 2021; Zomorodian & Carlsson, 2005). While persistent homology software provides topological insights, it can also provide some geometrical information (Bubenik et al., 2020). It would neither be appropriate to directly compare persistent homology with our approach, nor provide any detailed benchmarking, and so, we concern ourselves specifically with the global topology of manifolds and the estimation of Betti numbers, and only explore some of the basic practical details of persistent homology software.

Otter et al. (2017) present a diverse set of benchmark experiments using a collection of persistent homology software packages, and real and synthetic data from a variety of fields,

Table 3. The dataset parameters and the unit radius ranges for each factor of the manifolds that were used to produce 4D cavities; α is expressed in radians.

Cube	Parameters	B^4	$S^1 \times B^3$	$S^2 \times B^2$	$S^1 \times S^1 \times B^2$
32^4	32000 samples	B^4: 1.25 to 2	S^1: 2.5 to 4	S^2: 2.5 to 4	S^1: 2.5 to 8
	1 toxel boundary		B^3: 1.25 to 2	B^2: 1.25 to 2	S^1: 2.5 to 4
	1 to 16 holes				B^2: 1.25 to 2
	5 unit spacing				α: 0 to $\pi/2$

including topology. Most of the datasets that were used were 2D or 3D, however one 8D dataset was considered. Ripser is regarded as one of the most memory and time efficient implementations for computing persistent homology of Vietoris-Rips complexes (Bauer, 2021; Otter et al., 2017). Previously, Otter et al. (2017) also developed modifications to the standard reduction algorithm, which made it possible to parallelise persistent homology computations, and has served as a foundation for implementations (Bauer et al., 2014) such as Flagser (Lütgehetmann et al., 2020). GUDHI (Maria et al., 2014) is another highly regarded package that is available for both Python and C++, and is capable of analysing data with a variety of complexes and filtrations. Benchmarking suggests that GUDHI is well-suited for point cloud data, when compared with other options (Otter et al., 2017). Some other notable packages include DIPHA (Bauer et al., 2014), and Cubical Ripser (Kaji et al., 2020).

Experiments were performed on an Alienware R11 workstation, with an Intel i9 CPU, 64GB RAM, and an NVIDIA GeForce RTX3090 24GB GPU, as well as an NVIDIA DGX Station, with an Intel Xeon E5-2698 v4 CPU, 256GB RAM, and four V100-32GB GPUs. By considering the toxel cube as a point cloud with integral coordinates, it was appropriate to use GUDHI to find suitable dataset parameters (Otter et al., 2017). Vietoris-Rips, alpha, and cubical (both filtered and single) (Wagner et al., 2012) complexes were considered, and were applied to a sample by using the subset of toxels with value 1, since these were associated with the cube, rather than a cavity. Execution times varied, and were partially dependent on the collective number of included toxels. When operating over a single cubical complex, which comprised only of cubical simplexes between contiguous voxels with a value equal to 1, the cubical complex was completely determined by the voxels themselves; in theory, any persistent homology software would have applied the algorithm to the same complex and yielded the same results.

We began with 8^4 toxel cubes, and then attempted to analyse 16^4, 32^4, 64^4, 96^4, and 128^4 samples. Since the data points possessed integral coordinates, GUDHI was capable of correctly detecting holes of any dimension, provided that their diameter was greater than 2 units, or equivalently, greater than the 'distance' between diagonal points of a 4D unit cube ($\sqrt{4}$ units). Otherwise, Betti number calculations would suffer as a result of there being insufficient resolution to describe a hole with such a small radius. This afforded the use of restrictions where appropriate, in order to support the algorithm. For example, the dimension up to which the algorithm was asked to analyse was capped at 4, and the integral coordinate structure meant that it was safe to assume that the maximum edge length of the simplices necessary to compute a correct result was 2 units.

We were unable to successfully analyse a 32^4 cube using Vietoris-Rips or alpha complexes with the Alienware R11 workstation, however, it was possible to analyse a 3D 128^3 cube, which we noted consisted of more points than the 32^4 cube. When completely describing a sample with a single (unfiltered) cubical complex, which seemed to be particularly well-suited to image-type data such as ours (Otter et al., 2017), we were able to analyse a 64^4 cube, but not one of size 96^4. The same experiments were performed on the NVIDIA DGX Station, with which we were able to analyse a 96^4 cube using a cubical complex, but not one of size 128^4. Thus, the limits of our hardware was found to be somewhere between samples of sizes 96^4 and 128^4. These observations were most likely related to the requirements of the persistent homology algorithm itself, the computing resources required, and the complication of constructing, and working with, exponentially growing 4D complexes.

In order to handle samples on which it was not feasible to apply GUDHI directly, we experimented with firstly transforming the sample, via downsampling or using the cube's complement, in order to reduce the number of points fed into the algorithm. The first option caused issues stemming from a loss of resolution necessary to accurately describe holes. The second option required another step to undo the complementation, which was not one-to-one.

6. Convolutional neural networks for 4D data

Neural networks were implemented using the PyTorch machine learning framework (Paszke et al., 2019), however it was necessary to develop custom 4D CNN layers. We began with the convolution layer. For a sample x, filter f, and output channel $c_{out} \in C_{out}$, the 4D convolution operation was taken as a sum of the convolutions over each input channel $c_{in} \in C_{in}$, as expressed in Equation 4.

Three approaches to implementing the 4D convolution were investigated, each with user-definable kernel dimensions, padding, stride length, and GPU acceleration compatibility. An optional bias vector B was included, which contained a bias b_{out} for each output channel. Therefore, each approach computed an operation to the effect of $y[c_{out}, m, n, o, p] + b_{out}$.

- Naive approach: a direct application of the sum in Equation 4, using several nested loops.
- Reformulation approach: achieved by reformatting the sample and filter tensors into 2D matrices. The sample x was 'diced up' into sub-cubes x_i that each captured a location of convolution. The x_i were then vectorised, with their channels $x_{i,c_{in}}$ concatenated to form the ith row $[x_{i0}^T : x_{i1}^T : \cdots : x_{i(|C_{in}|-1)}^T]$ of the input matrix. The filter f was similarly reformatted into a matrix, with the jth column $[f_{0j}^T : \cdots : f_{(|C_{in}|-1)j}^T]^T$ comprising of a vectorisation of the kernel associated with the jth output channel. The convolution of the entire sample was then computed as the product of these two matrices (see Equation 5), and then appropriately reshaped for output.
- Extension approach: an implementation over PyTorch's native 3D convolution operation, similar to the approach used by Gessert et al. (2020) and Zhang et al. (2020). This approach is essentially a rearrangement of Equation 4, which is afforded since the sum is finite in practice. This modification results in a sum of 3D convolutions over the remaining dimension (indexed by l).

6.1. Convolution performance

We performed a series of pilot tests using the Alienware R11 workstation, with an aim to gauge how useful each implementation was. Relative speed and approximate batch size were our main concerns. Chosen somewhat arbitrarily, each implementation received an 8-channelled sample, convolved over the sample with a 4^4 unit kernel, and then output a 16-channelled result. Beginning with a random 8^4 4D tensor, a copy of this tensor was fed into each implementation, and the execution time of each was recorded. This was repeated with nine more random tensors, giving a total of ten records per implementation, which were then averaged to fill the first column of numbers in Table 4. The same tests were attempted with 16^4, 32^4, and 64^4 tensors. Not all tests were possible, and those that failed to execute, or finish in a reasonable time (less than 5 minutes), are signified with a hyphen (-). The naive approach failed to operate on larger samples, which was due to the inefficiencies of nested loops. The reformulation approach was significantly faster than the other two options, however, the trade-off was that it required much more memory to handle the reformatted data. For example, over 100GB of GPU memory was required to convolve a 64^4 sample. It also failed to convolve over a batch of more than two 32^4 samples. The extension approach was not the fastest, but demonstrated a better capability to handle multi-channelled batches, and was ultimately selected for our experiments.

Table 4. Summary of execution times (seconds). Tests that failed to execute, or finish in a reasonable time (less than 5 minutes), are signified with a hyphen (-).

Size	8^4	16^4	32^4	64^4
Naive	0.7374	36.452	-	-
Reform	0.0004	0.0013	0.0064	-
Extension	0.0074	0.0114	0.0322	0.5624

The batch size that we employed for our full scale experiments was dependent on available memory and the size of our network, which is discussed further in Section 7.

6.2. Pooling layers

In light of the 4D convolution performance observations, both 4D max pooling and average pooling layers were also implemented over their 3D analogues. The mathematical arguments allowing this were similar to the convolution case. Each implementation preserved PyTorch's default stride length, which matched the dimensions of the pooling kernel, however, these parameters were also user-definable.

7. Experiments and results

This project culminated in the design and training of a 4D CNN that was capable of learning to estimate the Betti numbers of our synthetic dataset. Our model was simple in design, and reflected those used by Paul & Chalup (2019). The CNN began with three iterations of a module consisting of a 4D convolution layer, followed by a ReLU function, and then a 4D max pooling layer.

Importantly, the first convolutional layer received the same 1-channelled data as GUDHI; the data was not downsampled before training. The convolutional layers output 8, 16 and 32 channels, respectively, and each utilised a 5^4 kernel, and a padding of 1 unit. The pooling kernel was 2^4 units. After the three convolution modules, the result was flattened, and then passed through two fully connected layers that were separated by a ReLU operation. Finally, the result was sparsely coded; one output neuron was reserved for each possible value of Betti number n. Based on the design of

$$y[c_{out}, m, n, o, p] = \sum_{c_{in} \in C_{in}} \sum_{l=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[c_{in}, i, j, k, l] \cdot f[c_{out}, c_{in}, m+i, n+j, o+k, p+l] \quad (4)$$

$$\begin{bmatrix} x_{00}^T : x_{01}^T : \cdots : x_{0(|C_{in}|-1)}^T \\ \vdots \\ x_{i0}^T : x_{i1}^T : \cdots : x_{i(|C_{in}|-1)}^T \\ \vdots \end{bmatrix} \begin{bmatrix} f_{00} & \cdots & f_{0(|C_{out}|-1)} \\ \vdots & \vdots & \vdots \\ f_{(|C_{in}|-1)0} & \cdots & f_{(|C_{in}|-1)(|C_{out}|-1)} \end{bmatrix} \quad (5)$$

our dataset, for Betti number 0, we accommodated for 0 and 1, and for Betti numbers 1 to 3, we accommodated for 0 to 8. Thus, the result was output to a vector of length $2 + 9 + 9 + 9 = 29$.

While preliminary testing was performed on an Alienware R11 workstation, we were fortunate to be able to deploy our full-scale deep learning experiments on an NVIDIA DGX Station. A multi-GPU arrangement with NVLink provided 128GB of GPU memory (via four V100-32GB GPUs), which accommodated the CNN, and allowed for a 128 sample batch size.

The dataset was randomly divided into 90% training, 5% validation, and 5% test sets at the beginning of each experiment. For each epoch, the samples were randomly rotated in multiples of 90 degrees through a randomly selected coordinate plane as they were fed into the Pytorch dataloader, which offered twelve possible variations on each sample. The Cross Entropy loss function was appropriately set up to handle the four separate outputs for β_0, β_1, β_2, and β_3. The Adam optimiser was initialised with a learning rate of 0.001, and a scheduler was employed to reduce the learning rate by a factor of 10 at epochs 160 and 180 over a 200 epoch training schedule.

Table 5 presents the test set accuracy average μ and standard deviation σ that were achieved in five repeats of the experiment. These figures can be taken as a proof-of-concept of the 4D CNN approach to Betti number estimation. We highlight that each experiment utilised randomly selected training, validation, and test sets, and required just over 4 days (approximately 97.5 hours) to complete each training schedule. An average combined test set accuracy of 94.66% was achieved.

It was observed that the number of epochs that were required to achieve a reasonable accuracy increased as the dimension of the Betti number increased. This was evident in full scale experiments, trial runs, and preliminary experiments, which employed CNNs that were trained to specialise in estimating a single Betti number. Our observation could be attributed to the diverse assortment of fundamental 3D holes versus the set of fundamental 2D holes (formed by B^3 and $S^1 \times B^2$) and 1D holes (formed by B^2). This would suggest that training a network to estimate higher dimensional Betti numbers may require larger datasets and longer training schedules. It would also warrant the use of CNNs that specialise in a subset of Betti numbers, when only a subset are of interest.

Table 5. Summary of CNN test set accuracies

Run	β_0	β_1	β_2	β_3	Combined
1	100	97.54	93.99	86.18	94.43
2	100	97.42	94.65	85.58	94.41
3	100	98.50	95.49	88.46	95.61
4	100	98.38	94.59	90.99	95.99
5	100	97.66	88.76	85.10	92.88
μ	100	97.90	93.50	87.26	94.66
σ	0	0.50	2.70	2.45	1.22

While the CNN approach was not as precise as the deterministic, and perhaps more general, persistent homology approach discussed in Section 5, the 4D CNN did demonstrate a potential to accurately analyse larger samples. However, since CNNs were inspired by, and model, our understanding of the neuronal organisation of the visual cortex (Nonaka et al., 2020; Xu & Vaziri-Pashkam, 2021), it is possible that the CNN approach is subject to similar limitations that humans face when using visual means to determine the homology of spaces with less than 4 dimensions, as opposed to using combinatorial devices, such as simplices.

8. Conclusion

It is evident from the results that the implemented 4D CNNs are capable of extracting patterns from our data to accurately infer Betti numbers, and demonstrate that a computer vision approach is robust enough, even when faced with the variability of cavity size, shape, and orientation seen in 4D. One possible progression could be the application of connected sums of objects and further homeomorphic deformations to introduce more diversity. Our investigation and some existing literature (Paul & Chalup, 2019; Zhou et al., 2021) suggest that if realistic synthetic data can be acquired, then a computer vision approach may be a suitable alternative in cases where using existing persistent homology software may be infeasible.

It was apparent from our tests that GUDHI, and perhaps persistent homology algorithms in general, did not tolerate downsampling as a means to mitigate the limitation faced when computing the Betti numbers of larger 4D samples. While the present study was limited to basic simulated data with low resolution to allow processing on our workstations within a sensible timeframe, it should be possible to process larger data samples or application data in the near future, or with larger computers. Solomon et al. (2022) show that subsampling methods can be effective when used to compute an averaged persistence image, and indicate the various intricacies of subsampling that may need to be covered. In parallel work (Hannouch & Chalup, 2023a), our aim was to extend the approach of the present paper to a dataset consisting of 128^4 samples with up to 48 holes of various dimensions. Directly analysing this data with our hardware was not possible due to the large memory and computational demands of applying persistent homology and CNNs to data of this size. Downscaling to a 32^4 resolution affected the topology of the data samples. However, the 4D CNN could still recover the topology of the original data with about 80% accuracy while GUDHI was not able to do so.

Finally, it could be asked if our CNNs have learned to see topological properties, such as 1-, 2- or 3-dimensional holes in 4D, similar to how a human can distinguish the topology of a doughnut from that of an apple in 3D. While we would like to leave the answer open, we note that for humans, it can already be a challenge to visually understand the topology of complex structures in 3D. The ability to visualise 3-manifolds in dimensions greater than 3 is a rare skill that may be mostly reserved to well-trained geometric topologists. Our brief excursion into algebraic topology in Section 4.1 indicates the mathematical machinery required to deal with some of the simplest objects. While persistent homology seems to run into complexity issues, it is quite astonishing to see what the basic CNN concept can achieve in the 4D context. By scaling up data complexity and training regimes, future 4D CNNs may become part of a topological data analysis tool set, which may then allow users to approach real-world applications with non-trivial underlying manifolds in 4D.

Acknowledgements

We would like to acknowledge support for this project through the Australian Research Council Discovery Project Grant DP210103304. The first author was supported by a PhD scholarship from the University of Newcastle associated with this grant. The authors are grateful for discussions or support to Asad Abbas, Nan Li, Matthew Muller, Dylan Peek, Ysobel Sims and Matthew Skerritt.

References

Adams, H., Tausz, A., and Vejdemo-Johansson, M. JavaPlex: A research software package for persistent (co)homology. In Hong, H. and Yap, C. (eds.), *Mathematical Software – ICMS 2014*, pp. 129–136, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44199-2.

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., and Ziegelmeier, L. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017. URL `http://jmlr.org/papers/v18/16-337.html`.

Al-Sahlani, K., Broxtermann, S., Lell, D., and Fiedler, T. Effects of particle size on the microstructure and mechanical properties of expanded glass-metal syntactic foams. *Materials Science and Engineering: A*, 728: 80 – 87, 2018. ISSN 0921-5093. URL `https://doi.org/10.1016/j.msea.2018.04.103`.

Ando, I., Mugita, Y., Hirayama, K., Munetoh, S., Aramaki, M., Jiang, F., Tsuji, T., Takeuchi, A., Uesugi, M., and Ozaki, Y. Elucidation of pore connection mechanism during ductile fracture of sintered pure iron by applying persistent homology to 4d images of pores: Role of open pore. *Materials Science and Engineering: A*, 828:142112, 2021.

Aziz, F., Wong, A. S., and Chalup, S. Semi-supervised manifold alignment using parallel deep autoencoders. *Algorithms*, 12(9):186, 2019. URL `https://doi.org/10.3390/a12090186`.

Bauer, U. Ripser: efficient computation of vietoris–rips persistence barcodes. *Journal of Applied and Computational Topology*, 5:1–33, 2021.

Bauer, U., Kerber, M., and Reininghaus, J. Distributed computation of persistent homology. In *2014 Proceedings of the Sixteenth workshop on Algorithm Engineering and Experiments (ALENEX)*, pp. 31–38. SIAM, 2014.

Bessieres, L., Besson, G., Boileau, M., Maillot, S., and Porti, J. *Geometrisation of 3-Manifolds*, volume 13 of *EMS Tracts in Mathematics*. European Mathematical Society, Zurich, 2010. URL `https://www-fourier.ujf-grenoble.fr/~besson/book.pdf`.

Brasselet, J.-P., Seade, J., and Suwa, T. The case of manifolds. In *Vector fields on Singular Varieties*, chapter 1, pp. 1–29. Springer, Berlin, Heidelberg, 2009. ISBN 9783642052057. URL https://doi.org/10.1007/978-3-642-05205-7_1.

Bubenik, P., Hull, M., Patel, D., and Whittle, B. Persistent homology detects curvature. *Inverse Problems*, 36(2): 025008, jan 2020. URL https://dx.doi.org/10.1088/1361-6420/ab4ac0.

Carlsson, G., Ishkhanov, T., de Silva, V., and Zomorodian, A. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76(1):1–12, 2008. URL https://doi.org/10.1007/s11263-007-0056-x.

Choy, C., Gwak, J., and Savarese, S. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3075–3084, 2019.

Clark, D. and Badea, C. Convolutional regularization methods for 4D, X-ray CT reconstruction. In *Medical Imaging 2019: Physics of Medical Imaging*, volume 10948, pp. 10948 2A. International Society for Optics and Photonics, 2019.

de Surrel, T., Hensel, F., Carrière, M., Lacombe, T., Ike, Y., Kurihara, H., Glisse, M., and Chazal, F. Ripsnet: a general architecture for fast and robust estimation of the persistent homology of point clouds. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022. URL https://openreview.net/forum?id=SeZlCN-ypgq.

Duarte, I., Fiedler, T., Krstulović-Opara, L., and Vesenjak, M. Cellular metals: Fabrication, properties and applications. *Metals*, 10(11):1545, 2020.

Edelsbrunner, H. and Harer, J. *Computational Topology: An Introduction*. Applied Mathematics. American Mathematical Society, 2010. ISBN 9780821849255.

Gallier, J. and Xu, D. *A Guide to the Classification Theorem for Compact Surfaces*, volume 9. Springer-Verlag, Berlin Heidelberg, 2013. ISBN 978-3-642-34363-6.

Gao, D., Chen, J., Dong, Z., and Lin, H. Connectivity-guaranteed porous synthesis in free form model by persistent homology. *Computers & Graphics*, 106:33–44, 2022. ISSN 0097-8493. URL https://www.sciencedirect.com/science/article/pii/S0097849322000917.

Gessert, N., Bengs, M., Schlüter, M., and Schlaefer, A. Deep learning with 4D spatio-temporal data representations for oct-based force estimation. *Medical Image Analysis*, 64: 101730, 2020.

Graham, B. Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*, 2014.

Graham, B. and van der Maaten, L. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

Hannouch, K. M. and Chalup, S. Topology estimation of simulated 4D image data by combining downscaling and convolutional neural networks. *arXiv preprint arXiv:2306.14442*, 2023a. URL https://doi.org/10.48550/arXiv.2306.14442.

Hannouch, K. M. and Chalup, S. Supplementary material of the paper: Learning to see topological properties in 4D using convolutional neural networks, 2023b. URL http://dx.doi.org/10.25817/GMW8-Q849.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. URL https://doi.org/10.1038/s41586-020-2649-2.

Hatcher, A. *Algebraic Topology*. Cambridge University Press, Cambridge, UK, 2002.

Hofmann, L., Rieck, B., and Sadlo, F. Visualization of 4D vector field topology. *Computer Graphics Forum*, 37(3): 301–313, 2018. doi: https://doi.org/10.1111/cgf.13421. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13421.

Kaji, S., Sudo, T., and Ahara, K. Cubical ripser: Software for computing persistent homology of image and volume data. *arXiv preprint arXiv:2005.12692*, 2020.

Kassab, L., Howland, S., Kvinge, H., Kappagantula, K. S., and Emerson, T. TopTemp: Parsing precipitate structure from temper topology. In Cloninger, A., Doster, T., Emerson, T., Kaul, M., Ktena, I., Kvinge, H., Miolane, N., Rieck, B., Tymochko, S., and Wolf, G. (eds.), *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022*, volume 196 of *Proceedings of Machine Learning Research*, pp. 199–205. PMLR, 25 Feb–22 Jul 2022. URL https://proceedings.mlr.press/v196/kassab22a.html.

Kaufman, E. L., Lord, M. W., Reese, T. W., and Volkmann, J. The discrimination of visual number. *The American Journal of Psychology*, 62(4):498–525, 1949.

Kim, D., Wang, N., Ravikumar, V., Raghuram, D., Li, J., Patel, A., Wendt III, R. E., Rao, G., and Rao, A. Prediction of 1p/19q codeletion in diffuse glioma patients using pre-operative multiparametric magnetic resonance imaging. *Frontiers in Computational Neuroscience*, 13: 52, 2019.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, pp. 1097–1105. Curran Associates, Inc., 2012.

Lee, J. M. *Introduction to Topological Manifolds*. Springer-Verlag, 2000.

Loughrey, C. F., Fitzpatrick, P., Orr, N., and Jurek-Loughrey, A. The topology of data: opportunities for cancer research. *Bioinformatics*, 37(19):3091–3098, 2021.

Lütgehetmann, D., Govc, D., Smith, J. P., and Levi, R. Computing persistent homology of directed flag complexes. *Algorithms*, 13(1):19, 2020.

Maria, C., Boissonnat, J., Glisse, M., and Yvinec, M. The GUDHI library: Simplicial complexes and persistent homology. In *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, volume 8592 of *Lecture Notes in Computer Science*, pp. 167–174. Springer, 2014. URL https://doi.org/10.1007/978-3-662-44199-2_28.

MATLAB. *version 9.3.0 (R2017b)*. The MathWorks Inc., Natick, Massachusetts, 2017.

Myronenko, A., Yang, D., Buch, V., Xu, D., Ihsani, A., Doyle, S., Michalski, M., Tenenholtz, N., and Roth, H. 4D CNN for semantic segmentation of cardiac volumetric sequences. In *International Workshop on Statistical Atlases and Computational Models of the Heart 2019*, pp. 72–80. Springer, 2019.

Nonaka, S., Majima, K., Aoki, S. C., and Kamitani, Y. Brain hierarchy score: Which deep neural networks are hierarchically brain-like? *bioRxiv*, 2020. URL https://www.biorxiv.org/content/early/2020/07/25/2020.07.22.216713.

Noor, M. B. T., Zenia, N. Z., Kaiser, M. S., Al Mamun, S., and Mahmud, M. Application of deep learning in detecting neurological disorders from magnetic resonance images: a survey on the detection of alzheimer's disease, parkinson's disease and schizophrenia. *Brain Informatics*, 7(1):1–21, 2020.

Otter, N., Porter, M. A., Tillmann, U., Grindrod, P., and Harrington, H. A. A roadmap for the computation of persistent homology. *EPJ Data Science*, 6:17–55, 2017. URL https://doi.org/10.1140/epjds/s13688-017-0109-5.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Paul, R. and Chalup, S. Estimating Betti numbers using deep learning. In *2019 International Joint Conference on Neural Networks (IJCNN 2019)*. IEEE, 2019. URL 10.1109/IJCNN.2019.8852277.

Reimann, M. W., Nolte, M., Scolamiero, M., Turner, K., Perin, R., Chindemi, G., Dlotko, P., Levi, R., Hess, K., and Markram, H. Cliques of neurons bound into cavities provide a missing link between structure and function. *Frontiers in Computational Neuroscience*, 11:48, 2017. ISSN 1662-5188. URL https://www.frontiersin.org/article/10.3389/fncom.2017.00048.

Saadatfar, M., Takeuchi, H., Robins, V., Francois, N., and Hiraoka, Y. Pore configuration landscape of granular crystallization. *Nature Communications*, 8(1):1–11, 2017.

Sagristà, A., Jordan, S., Just, A., Dias, F., Nonato, L. G., and Sadlo, F. Topological analysis of inertial dynamics. *IEEE Transactions on Visualization and Computer Graphics*, 23 (1):950–959, 2017. doi: 10.1109/TVCG.2016.2599018.

Scott, R. B., Pontin, D. I., and Wyper, P. F. Magnetic structures at the boundary of the closed corona: A semi-automated study of s-web morphology. *The Astrophysical Journal*, 882(2):125, sep 2019. URL https://dx.doi.org/10.3847/1538-4357/ab364a.

Solomon, E., Wagner, A., and Bendich, P. From Geometry to Topology: Inverse Theorems for Distributed Persistence. In Goaoc, X. and Kerber, M. (eds.), *38th International Symposium on Computational Geometry (SoCG 2022)*, volume 224 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 61:1–61:16, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-227-3. doi: 10.4230/LI

PIcs.SoCG.2022.61. URL https://drops.dagstuhl.de/opus/volltexte/2022/16069.

Som, A., Choi, H., Ramamurthy, K. N., Buman, M. P., and Turaga, P. 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*, pp. 3639–3648, 2020.

Stolz, B. J., Emerson, T., Nahkuri, S., Porter, M. A., and Harrington, H. A. Topological data analysis of task-based fmri data from experiments on schizophrenia. *Journal of Physics: Complexity*, 2(3):035006, may 2021. URL https://dx.doi.org/10.1088/2632-072X/abb4c6.

Wagner, H., Chen, C., and Vuçini, E. Efficient computation of persistent homology for cubical data. In Peikert, R., Hauser, H., Carr, H., and Fuchs, R. (eds.), *Topological methods in data analysis and visualization II*, pp. 91–106. Springer, 2012.

Xu, Y. and Vaziri-Pashkam, M. Limits to visual representational correspondence between convolutional neural networks and the human brain. *Nature Communications*, 12(2065), 2021. URL https://doi.org/10.1038/s41467-021-22244-7.

You, Q. and Jiang, H. Action4D: Real-time action recognition in the crowd and clutter. *arXiv preprint arXiv:1806.02424*, 2018.

Zhang, L., Wang, L., Yang, B., Niu, S., Han, Y., and Oh, S.-K. Rapid construction of 4D high-quality microstructural image for cement hydration using partial information registration. *Pattern Recognition*, 124:108471, 2022. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.108471. URL https://www.sciencedirect.com/science/article/pii/S0031320321006476.

Zhang, S., Guo, S., Huang, W., Scott, M. R., and Wang, L. V4D: 4D convolutional neural networks for video-level representation learning. *arXiv preprint arXiv:2002.07442*, 2020.

Zhou, C., Dong, Z., and Lin, H. Learning persistent homology of 3D point clouds. In *Computers & Graphics: Special Section on Shape Modeling International (SMI) 2021*, volume 101. Elsevier, 2021. doi: https://doi.org/10.1016/j.cag.2021.10.022. URL https://www.sciencedirect.com/science/article/pii/S0097849321002399.

Zomorodian, A. and Carlsson, G. Computing persistent homology. *Discrete & Computational Geometry*, 33(2): 249–274, 2005. URL https://doi.org/10.1007/s00454-004-1146-y.

A. Visualising selected 3-manifolds

Visualisations of the basic 3-manifolds that were studied in this work were created with MATLAB (MATLAB, 2017) using an (x, y, z, w)-system. 3D slices were taken along the w-axis, and ordered from left to right, with the central image depicting the slice at $w = 0$. Just as we may see a circle, or pairs of circles, when taking 2D slices of $S^1 \times S^1$, we may see a circle inflate into a torus, which then splits into two tori that then collapse back into one torus, and then return to a circle when using this approach to visualise $S^1 \times S^1 \times S^1$ (as in Figure 1). A cut-away was used in Figures 4 and 5 in order to expose the *inner* surface that would otherwise have been hidden by the *outer* enveloping surface. For example, Figure 5 depicts $S^1 \times S^1 \times S^1$, with $\alpha = \pi/2$, which begins as a torus, and then splits into two concentric tori, before returning to a single torus. Similarly, when depicting $S^2 \times S^1$, we see a sphere that splits into two concentric spheres, which then join together again (Figure 4).

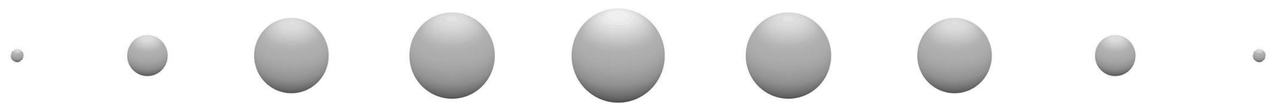

Figure 2. Visualising S^3 along the w-axis.

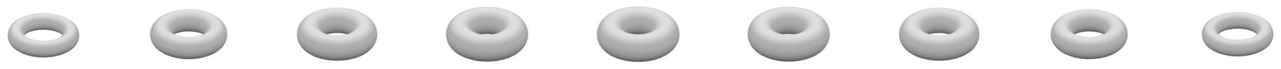

Figure 3. Visualising $S^1 \times S^2$ along the w-axis.

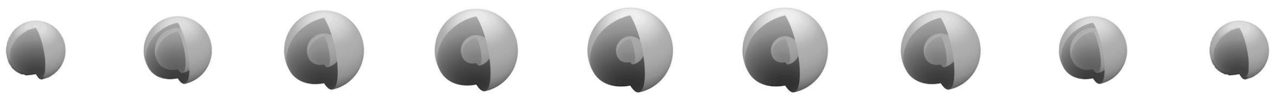

Figure 4. Visualising $S^2 \times S^1$ along the w-axis.

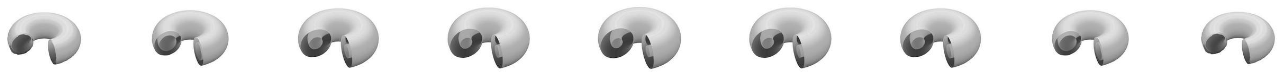

Figure 5. Visualising $S^1 \times S^1 \times S^1$ with $\alpha = \pi/2$ along the w-axis.

B. Visualising the dataset

By inverting the toxel values (setting 0 to 1, and vice versa), we are able to visualise the cavities within a sample. In Figure 6, we inspect a 32^4 sample by selecting 16 equally-spaced 3D slices while traversing along the w-axis. The slices are ordered from left to right and top to bottom. In this particular sample, we can see at least one example of each cavity described in Section 4.2. We begin to see $S^1 \times B^3$ in the second slice, followed by $S^1 \times S^1 \times B^2$ and a ball in the fourth slice. In slice nine, we begin to see $S^2 \times B^2$. Note that these examples look quite different to the visualisations that are included in Appendix A; what we see when visualising a 4D sample depends on how the cavities are oriented within the sample, and the perspective that we decide to inspect the sample from. The associated label encodes $\beta_0 = 1$, $\beta_1 = 2$, $\beta_2 = 3$, and $\beta_3 = 6$. A video showing all 32 slices is available from Hannouch & Chalup (2023b).

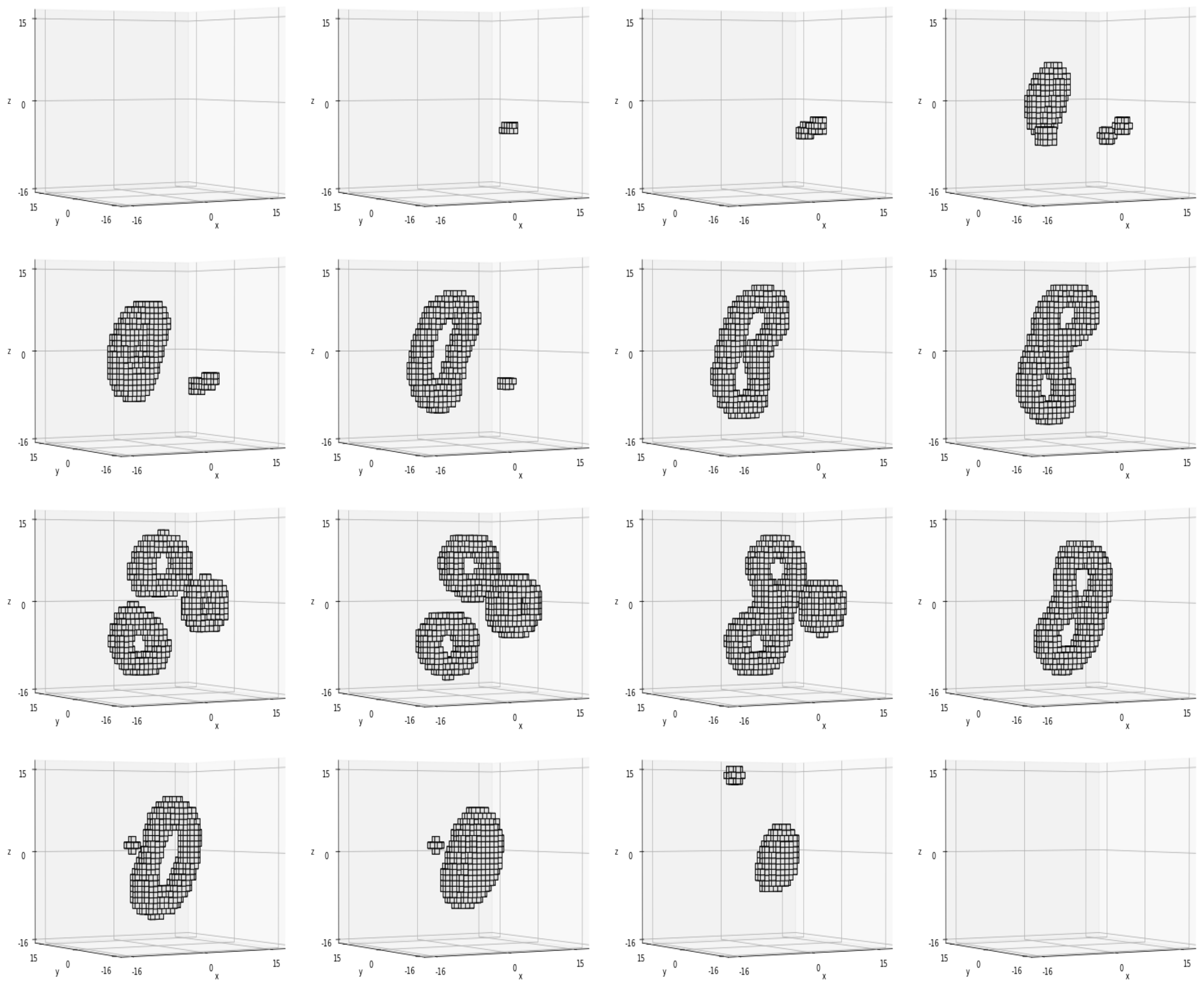

Figure 6. By inverting the toxel values (setting 0 to 1, and vice versa), we are able to visualise the cavities within a sample. Here, we inspect a 32^4 sample by selecting 16 equally-spaced 3D slices while traversing along the w-axis. The slices are ordered from left to right and top to bottom.

C. Notes on algebraic topology

A general introduction to algebraic topology that can serve as background to our approach can be found in the books of Edelsbrunner & Harer (2010) and Hatcher (2002). In this section, we elaborate on some of the concepts that were referenced in Section 4.1 of the paper.

C.1. Reduced Homology

In contrast to the higher dimensional Betti numbers, β_0 is often interpreted as the number of connected components, rather than some number of holes. It would seem more consistent if $\beta_0 = 1$ implied the existence of a 'gap' between two components; this is the rationale behind using reduced homology, and its application can simplify some calculations. The modification is achieved by introducing an augmentation map into the chain complex that is used in the derivation of homology theory, and the n^{th} *reduced* homology group is often denoted $\tilde{H}_n$. The pth reduced Betti number is denoted $\tilde{\beta}_p$, and is analogously defined as $\operatorname{rank}\tilde{H}_p$. The effect of these changes is that $\tilde{\beta}_p = \beta_p$ for all $p > 0$, and $\tilde{\beta}_0 = \beta_0 - 1$, as desired. The reader is directed to Hatcher (2002, Chapter 2) for more.

C.2. The Mayer-Vietoris sequence

We state two versions of the Mayer-Vietoris sequence in singular homology. A proof can be found in Edelsbrunner & Harer (2010, Chapter 4.4), and more discussion can be found in Hatcher (2002, Chapter 2.2).

Let X be a topological space, and A and B be two subspaces whose interiors cover X; the interiors of A and B may intersect. The Mayer-Vietoris sequence is a long exact sequence that relates the singular homology groups (with coefficient group $\mathbb{Z}$) of X, A, B, and $A \cap B$ by

$$\begin{aligned} \cdots \to H_{n+1}(X) \xrightarrow{\partial_{n+1}} H_n(A\cap B) \xrightarrow{(i_n, j_n)} H_n(A) \oplus H_n(B) \xrightarrow{k_n - l_n} \\ H_n(X) \xrightarrow{\partial_n} H_{n-1}(A\cap B) \to \cdots \to H_0(A) \oplus H_0(B) \xrightarrow{k_0 - l_0} H_0(X) \to 0, \end{aligned} \tag{6}$$

where $i : A \cap B \to A$, $j : A \cap B \to B$, $k : A \to X$, and $l : B \to X$ are inclusion maps, $\oplus$ denotes the direct sum, and ∂_n denotes the n^{th} boundary homomorphism.

Assuming that the intersection of A and B is not empty, the Mayer-Vietoris sequence for reduced homology is identical to Equation 6 for $n > 0$, and ends with

$$\cdots \to \tilde{H}_0(A\cap B) \xrightarrow{(i_0, j_0)} \tilde{H}_0(A) \oplus \tilde{H}_0(B) \xrightarrow{k_0 - l_0} \tilde{H}_0(X) \to 0. \tag{7}$$

C.3. The Künneth theorem

The classical statement of the Künneth theorem for principal ideal domains, such as any field $\mathbb{F}$, or as in our case, the ring of integers $\mathbb{Z}$, relates the singular homology of two topological spaces X and Y with their product space $X \times Y$. The reader is directed to Hatcher (2002, Chapter 3.B) for a review of several versions of this theorem, and an explanation of the Tor functor.

Given a principal ideal domain R, and any topological spaces X and Y, the Künneth theorem states that there are short exact sequences, such that

$$0 \to \bigoplus_{i+j=k} H_i(X;R) \otimes_R H_j(Y;R) \to H_k(X \times Y;R) \to \bigoplus_{i+j=k-1} \operatorname{Tor}_1^R(H_i(X;R), H_j(Y;R)) \to 0, \tag{8}$$

where $\otimes_R$ denotes the tensor product.

D. Deriving an implicit formula

As explained in the paper, an implicit representation of $S^1 \times S^1 \times S^1$ can be derived by rotating the torus $S^1 \times S^1$ about S^1. To demonstrate this, we begin with a parametric equation for the torus with an arm radius a, and a centre radius r (distance from the origin to the centre of its arm)

$$(X, Y, Z) = ((r + a\cos\theta)\cos\phi, (r + a\cos\theta)\sin\phi, a\sin\theta). \tag{9}$$

A rotation about the y-axis can be performed with the matrix

$$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}, \tag{10}$$

and if we let $A = \cos\alpha$ and $B = \sin\alpha$, then applying $R_y(\alpha)$ to Equation 9 results in a 'tilted' torus described by $(AX + BZ, Y, -BX + AZ)$. By positioning this torus at $(0, 0, R)$, and then rotating it about the xy-plane, we effectively incorporate the remaining S^1 factor to find the following parametric equation of $S^1 \times S^1 \times S^1$,

$$(x, y, z, w) = ((R - BX + AZ)\cos\gamma, (R - BX + AZ)\sin\gamma, Y, AX + BZ). \tag{11}$$

Since $\sqrt{x^2 + y^2} - R = -BX + AZ$, and $A^2 + B^2 = 1$, it follows that

$$\begin{aligned}\sqrt{(-B(\sqrt{x^2+y^2}-R)+Aw)^2+z^2}-r &= \sqrt{(B^2X - BAZ + A^2X + BAZ)^2 + Y^2} - r \\ &= \sqrt{((A^2+B^2)X)^2+Y^2}-r = \sqrt{X^2+Y^2}-r = a\cos\theta.\end{aligned} \tag{12}$$

Similarly, we find that

$$A(\sqrt{x^2+y^2}-R)+Bw = -ABX + A^2Z + ABX + B^2Z = (A^2+B^2)Z = Z = a\sin\theta. \tag{13}$$

Equations 12 and 13 imply that $S^1 \times S^1 \times S^1$ can be described by

$$(\sqrt{(-B(\sqrt{x^2+y^2}-R)+Aw)^2+z^2}-r)^2 + (A(\sqrt{x^2+y^2}-R)+Bw)^2 = a^2. \tag{14}$$

The remaining formulas listed in Table 2 may be derived in a similar way.

E. Dataset description

A dataset supplement to this paper is available from Hannouch & Chalup (2023b). The repository provides a subset of data `data.zip` for demonstration, as well as the full 192.3 MB dataset that was used in this work.

F. Guide to code

PyTorch (Paszke et al., 2019) implementations of the 4D convolution and pooling layers that were described in the paper are available from Hannouch & Chalup (2023b), and can be found in the `src` folder, along with a script that trains our 4D CNN using the dataset found in `data.zip`. We have also provided three pre-trained models, and a small random dataset `test_data.zip` on which to demonstrate inference.

F.1. Specification of dependencies

The code is self-contained and executable, and should be run on a system with CUDA compatible GPUs. The Python requirements can be found in the `requirements.txt` file, and may be installed using:

```
> pip3 install -r requirements.txt
```

F.2. Data preparation

The `test_data.zip` file can be extracted to a folder named `test_data` using the command:

```
> unzip test_data.zip -d test_data
```

The `data.zip` file can be extracted to a folder named `data` using:

```
> unzip data.zip -d data
```

F.3. Training code

The 4D CNN described in this paper is implemented in `cnn4d.py`, and may be trained with the dataset found in the `data` folder using the following command. The GPUs that are employed for training are specified with the last argument, for example `"[0]"`.

```
> python3 train.py data "[0]"
```

The resulting model will be saved to the working directory as `demo_saved_model_<accuracy>`.

F.4. Evaluation code

The following command loads a pre-trained model `saved_model_1` found in the `models` folder, and performs inference on the data found in the `test_data` folder.

```
> python3 eval.py test_data "[0]" saved_model_1
```

The output lists the accuracy with which the network has estimated Betti numbers 0 to 3.