Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Birk Dybesland Ramberg

Topological Data Analysis and the Activations in a Convolutional Neural Network

Master's thesis in Industrial Mathematics
Supervisor: Marius Thaule
Co-supervisor: Melvin Vaupel
December 2021

NTNU
Norwegian University of Science and Technology

Birk Dybesland Ramberg

Topological Data Analysis and the Activations in a Convolutional Neural Network

Master's thesis in Industrial Mathematics
Supervisor: Marius Thaule
Co-supervisor: Melvin Vaupel
December 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

TOPOLOGICAL DATA ANALYSIS AND THE ACTIVATIONS IN A CONVOLUTIONAL NEURAL NETWORK

BIRK D. RAMBERG

Abstract. Topological Data Analysis (TDA) is concerned with studying the shape of data. TDA can extract valuable qualitative and quantitative information from complicated datasets. Convolutional Neural Networks (CNNs) have state-of-the-art performance on a variety of tasks, but are not fully understood. Through studying the activations of neurons at an intermediate layer in a CNN, we can gain a better understanding, both of a particular CNN, and of CNNs in general. A set of activations constitute an interesting, but complex dataset. In this thesis, I review a number of TDA methods which is then applied to the dataset of activations. This leads to what I call the Mapper Activation Atlas. I also investigate clusters of activations with persistent homology and UMAP, identifying and parametrising circular structure.

Sammendrag. Topologisk dataanalyse (TDA) omhandler studiet av datas form. TDA kan finne verdifull kvalitativ og kvantitativ informasjon fra kompliserte dataset. Convolutional Neural Networks (CNNs) løser et mangfold av oppgaver bedre enn noen annen teknologi, men er ikke fullstendig forstått. Gjennom å studere aktiveringer av nevroner i et mellomliggende lag i en CNN, kan vi oppnå en bedre forståelse, både av den spesifikke CNN, og av CNNs generelt. Et set av aktiveringer utgjør et interessant, men komplekst datasett. I denne oppgaven gjennomgår jeg et antall TDA-metoder, som senere blir anvendt på datasettet av aktiveringer. Dette leder til det jeg kaller the Mapper Activation Atlas. Jeg undersøker også klustere av aktiveringer med persistent homologi og UMAP, og identifiserer og parametriserer sirkelstruktur.

Acknowledgements

I would like to thank Magnus Bakke Botnan for his excellent introductory notes about TDA, and Rune Haugseng for neat notes on algebraic topology. I also thank Erik Rybakken for a piece of code for doing circular parametrisation.

Most deserving of praise are my supervisor Marius Thaule and PHD student Melvin Vaupel for their guidance, knowledge, patience, feedback, excitement and dedication.

I would like to thank my family and friends for their support. Lastly, I thank everyone who has listened patiently to me talking about topology, TDA or neural networks.

Date: December 23, 2021.

Contents

1. Introduction

Over the last 20–30 years, there has emerged methods designed to study the *shape* of data, seen through the lens of *topology*. Initially, the main topic was the *homology inference problem* [38]: How can we infer the topological invariant homology of a space from samples from the space? To solve this problem, the ideas of *persistence* were applied: In order to infer the homology, we study the change in the homology across different scales of the dataset. We can thus understand the dataset at different scales and are able to distinguish noise or local structure (which is highly scale dependent) from global structure (which is not as scale dependent). Through the work of Edelsbrunner et al. [15], Zomorodian and Carlsson [64] and Bauer [4], among others, *persistent homology* has become the main technique in the field of Topological Data Analysis (TDA).

Persistent homology has recently found applications within a broad variety of fields. For example, Perea et al. [39], [17] analyse (quasi-)periodic time series data by applying persistent homology to the sliding window embedding. Satisfyingly, they detect loop structure for periodic data and torus structure in quasi-periodic data. Moreover, persistent homology has found a number of applications in tandem with other machine learning methods [21], for example in the Topological Autoencoder [31], Neural Persistence [41] and topological regularisation of classifiers [11].

Later, perhaps inspired by the success of applying the theory of simplicial complexes and algebraic topology to data analysis, new topologically inspired methods like *Mapper* [50], *UMAP* [30] and *ToMaTo clustering* [10] have found a wide range of applications. For example, Mapper has been used to identify subgroups of diabetes type 2 [26] and to study medical data of cancer patients [28].

Most major, recent breakthroughs in machine learning have come from applications or enhancements of deep neural networks. For example, the Natural Language Processing (NLP) system GPT-3 [5] and the protein structure predictor AlphaFold [23] attracted massive attention in 2020. In particular, computer vision has become an especially suitable area for Convolutional Neural Networks (CNNs). The developments and improvements of CNNs in computer vision, and especially image recognition, have been fuelled by the ImageNet dataset [13] and the analogies with the brain. In addition, the wide range of useful applications of computer vision systems in everything from social media and image processing to autonomous vehicles has helped drive the development.

While the state-of-the-art performance of the systems speak for itself, the theory behind deep (convolutional) neural networks has not been able to keep up with the rapid improvements. One might say that we know that deep neural networks can perform complex tasks incredibly well, but we are not quite sure about why they work so well. For example, why does the heavy overparameterisation not lead to overfitting? The traditional approach in terms of generalisation has been to follow the principle of Occam's razor [60] : "Entities should not be multiplied beyond necessity". In machine learning, this translates to not allowing more freedom for the model than what is required for solving the task sufficiently. Deep learning is a breach of this paradigm. For example, in the paper exploring this topic [63], the authors find that state-of-the-art image classification CNNs can easily fit a random labelling of the images, illustrating the models' ability to express essentially any relationship. This is also confirmed by the theory of expressivity [18].

Moreover, the black box [55] nature of deep neural networks makes it difficult to distinguish a well-trained neural network from a poorly trained one. If we do not understand how the neural network works and what it has learned, we do not know what kinds of applications that would be suitable. In that case, we also do not understand when and why the network fails. This introduces a huge barrier to the applicability of the technology, despite its state-of-the-art performance. For many applications the risk associated with under-performance is too large to justify the introduction of a system that we can not understand, and therefore not trust. Hence, research aimed at understanding deep neural networks better is of great importance.

In particular, understanding CNNs has been a focus of such research. For example, significant research has been devoted to *feature visualisation*, creating visualisations of what different parts of the CNN look for. *Attribution* research makes methods for understanding why the network produced a particular output. For example, one could ask an image classification CNN which parts of an image that were relevant for making the classification that it did. In the paper [9], the authors create an Activation Atlas, combining ideas from feature visualisation and attribution research to create a global view of how the CNN sees a set of images.

Some research for understanding CNNs takes a more abstract approach. Carlsson and Gabrielsson [7] study the weights (arranged in spatial patches) of a CNN with methods from Topological Data Analysis. In the first layer they find a primary circle corresponding to edges of different orientation in the images, as well as a secondary and tertiary circle. Furthermore, they find the whole space of patches to lie on a Klein bottle, and illustrate that enforcing this behaviour of the weights in CNNs can be beneficial [27]. Their results are consistent with a similar analysis performed on patches in natural images [8] and on the visual cortex of monkeys [6]. This consistency suggests that studying the CNN can help to understand the brain (at least the parts responsible for vision) better, and vice versa. Moreover, the consistency with the space of image patches suggests that the weights of a CNN in a sense mimic the data it is given. The CNN only looks for things that actually exist in the images it is trained on.

Carlsson and Gabrielsson also apply their method of studying *spatial filters*, that is, filters that look only at spatial combinations of activations in the previous layer, not at combinations of activations across different channels. See Section 10 or [3] for details about CNNs. They find structure in the deeper layers of CNNs like vgg16 [49]. However, while this approach sheds much light on the weights in the first layer, where the spatial filters are what is of most interest, the approach has some drawbacks when applied to the deeper layers. Arguably, much of the power of CNNs comes from the ability to combine *different* features detected into more complex features in the subsequent layers. Considering the weights in the form of spatial filters allows only analysis of the networks ability to combine spatial patterns of a *single* channel. For example, their approach allows only to understand how the network combines *vertical edges* at different spatial locations, while the combination of *edges of all directions* is arguably more relevant.

One approach to address this challenge is to consider vectors of weights across channels instead of patches across spatial locations. This comes with the opposite weakness of not allowing for analysis of the spatial patters used to assemble different features into more complex features. However, since considering the weights in their

natural cuboid format is infeasible due to the high dimensionality ($\sim 10^5$), I will in this thesis focus on the vectors across channels.

When studying the brain, the "weights" of the neurons in the brain are of course not readily obtainable, as they are for a CNN. What we can (fairly) easily access are the *activations* in different neurons. This approach is taken by Rybakken et al. [43]. Working with neural activation data from mice, they detect a circular feature and find a correspondence with the head directions of the mice. This paper serves as an example of how one can reveal specific features about a complicated space to gain partial understanding. This conceptual approach will also be taken in this thesis.

In line with the observation that the CNNs "look for what exists", we will study the *space of activations* rather than the space of weights in this thesis. This places the work in this thesis closer to the work by Rybakken et al. [43] and Carter et al. [9] on Activation Atlases. However, the work by Carlsson et al. [8, 7, 6, 27] is of great relevance and inspiration.

1.1. **Outline.** This thesis is roughly divided into three parts. The first part (Sections 2 – 9) gives a brief introduction to a number of methods from *Topological Data Analysis*, with a focus on the intuition behind the methods. The second part (Sections 10 – 12) focuses on *convolutional neural networks*, methodologies for analysing the *space of activations* and results obtained with the methodologies. The last part (Section 13) consists of thoughts and discussions about potential future work.

In Section 2, the ideas of Topological Data Analysis are introduced and the problem that Topological Data Analysis tries to solve is formulated. In Section 3, we give a brief background on topology. We define *topological spaces*, *homeomorphisms* and *homotopy equivalences*. For more about the basics of topology I refer to [32]. In Section 4, we introduce *simplicial complexes*, an essential building block for the theory of Topological Data Analysis. In Section 5, we introduce *simplicial homology*, a topological invariant. I refer the reader to [33, Chapter 1] for more about simplicial complexes and simplicial homology. In Section 6, we explain *persistent homology*. Persistent homology is often described in the language of persistence modules and category theory, but we avoid these topics for accessibility. For this reason, the parts of the theory involving the decomposition of persistence modules, inference of the homology groups and stability of persistent homology are left out. I refer the reader to [14, Chapters VII and VIII], [36, Chapters 1–5] or my introduction to persistent homology [40] for the theory. We also include a subsection about preprocessing often applied to data before the computation of persistent homology. In Section 7, we review the method for obtaining a *circular parametrisation* from a computation of persistent (co)homology, as introduced in the paper [47]. In Section 8, we review *Mapper* [50]; a method that turns a dataset into a graph. In Section 9, I give a brief introduction to the manifold learning dimensionality reduction method *UMAP* [30], inspired by the description given on "How UMAP Works" from [25]. For the full theory and details of UMAP I refer the reader to the original paper [30].

In Section 10, we review *neural networks* and *convolutional neural networks*. We introduce the *space of activations* and discuss briefly *features visualisation* and *attribution*. Moreover, we discuss some of the work performed by Carlsson et al. on the weights of CNNs. For more about neural networks, I refer the reader to [3]. In Section 11, we review and discuss the Activation Atlas from [9], and propose a

new kind of activation atlas; the *Mapper Activation Atlas*. Some applications of the Mapper Activation Atlas are presented. In Section 12, we investigate clusters of the space of activations with a combination of the methods UMAP, persistent cohomology with circular parametrisation and Mapper to reveal and explain loop structures in subsets of the data.

Finally, in Section 13, I summarise and present ideas for further work that seem worthwhile pursuing.

2. Topological Data Analysis

We often want to study the *shape* of data, since the shape carries high-level information. Relevant questions to ask could be: Does the data divide into clusters, or does the data seem to be connected? If we can divide the data into clusters, what is the shape of the clusters themselves? Does the data lie on some manifold? Does the data exhibit any form of circularity or periodicity? Does the data have any flares?

These questions are of a topological nature and could therefore be investigated with tools from topology. After all, topology is the study of shape. When looking for the topological properties sketched above it is often advantageous to have methods that can disregard other properties of the data than those we are interested in. This might help to reveal the properties that we are truly interested in. In some ways one might say topology at its core is essentially about disregarding most properties of spaces in order to investigate what we are left with. This "disregarding" is done formally through the definition of topological spaces and the notions of *homeomorphisms* and *homotopy equivalences*. This is, in my opinion, the strongest argument for the use of topological methods on data.

2.1. Problem Description.

The problem we will address can usually be stated in a very general way. Let M be a metric space. We are given data, in the form of a point cloud, that is a finite set of points $P \subseteq M$, with a metric. We assume that the data are sampled from some topological space $K \subseteq M$, potentially with some noise. What can we say about the topology of the space K, and under what assumptions?

Remark 2.1. It is clear that we must assume some relationship between the point cloud P and the space K. Typical assumptions could be that we assume that P is sampled from K with some noise, or that K is a high density region of a probability density distribution that the samples in P are assumed drawn from.

The first problem we encounter is that P is a point cloud, so from a topological perspective P is simply $|P|$ disconnected points, with no interesting topological properties. We will overcome this challenge by associating more interesting topological spaces with the point cloud P. A simple example would be to consider the space we obtain by taking the union of balls of radius r, centered at all the points in P. See Figure 1. Under sensible assumptions, the union of balls shares much of the topology of the space we are really investigating, K.

3. Topological Spaces, Homeomorphisms and Homotopy Equivalences

The idea of *disregarding* properties of spaces in topology is formalised through the definition of *topological spaces* and the notions of *homeomorphisms* and *homotopy equivalences*. For example, in topology one would consider two circles with different

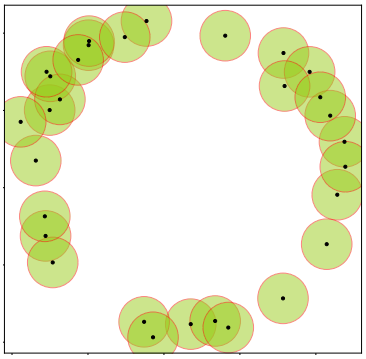


FIGURE 1. A point cloud and balls with fixed radius centered at the points.

radii to be essentially the same. A topologist would not even care whether the circle is actually a circle or an ellipse or any other kind of loop. The loop is, however, topologically different from a disk or a line segment. For a thorough introduction to topology see the book by Munkres [32].

We start by stating the formal definition of a topological space.

Definition 3.1. A **topological space** is a set X, endowed with a **topology** $\mathcal{T}$ on X. The topology $\mathcal{T}$ is a collection of subsets of X that are called *open*, and $\mathcal{T}$ satisfies the properties:

- The empty set $\emptyset$ and X are both in $\mathcal{T}$.
- Any union of subsets in $\mathcal{T}$ is also in $\mathcal{T}$.
- Any finite intersection of subsets in $\mathcal{T}$ is also in $\mathcal{T}$.

Remark 3.2. The above definition allows us to generalise the concept of continuity. However, in this thesis, we will not have to worry about the abstract properties of a topology since we will be working with the metric topology, which we will define shortly.

Example 3.3. Let X be any set. Then the power set $\mathcal{P}(X)$, that is the collection of all subsets of X, is a topology on X. This is called the *discrete topology* on X. All subsets of X are open.

The collection $\{\emptyset, X\}$ is also a topology on X. This is called the *indiscrete topology* on X. No subsets, except $\emptyset$ and X are open.

We will only deal with *metric spaces* in this thesis.

Definition 3.4. A **metric space** is a set X with a metric $d : X \times X \to \mathbb{R}$ satisfying, for any $x, y, z \in X$:

- $d(x, y) = 0$ if and only if $x = y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$

Theorem 3.5. *Let X be a metric space with metric d. Define $\mathcal{T}_d$ as the collection of subsets $U \subseteq X$ such that for any $x \in U$ there exists $r > 0$ such that $B(x; r) \subseteq U$, where $B(x; r)$ is the open ball with radius r centered at x and*

$$B(x; r) = \{y \in X : d(x, y) < r\}.$$

Then X is a topological space with topology $\mathcal{T}_d$. We call $\mathcal{T}_d$ the ***metric topology*** *on X induced by d.*

See the book by Munkres [32, Chapter 2] for proof.

Since all the spaces we are dealing with are metric spaces, when we say that a space is a topological space, we mean that it has the metric topology induced by the metric.

If X is a topological space and $U \subseteq X$ is a subset of X, there exists a topology on U that consists of the pairwise intersections of U with all open subsets of X. This is called the *subspace topology* on U.

If X and Y are topological spaces then there is an induced topology on the set $X \times Y$ that is called the *product topology*. See Munkres [32, Chapter 2].

If X is a topological space and $\sim$ an equivalence relation on the elements of X, there exists a topology on the set of equivalence classes $X/\sim$ called the *quotient topology*.

3.1. **Continuous Maps and Homeomorphisms.**

Definition 3.6. Let X and Y be topological spaces with topologies $\mathcal{T}_X$ and $\mathcal{T}_Y$, respectively. A map $f : X \to Y$ is **continuous** if for any open set $V \in \mathcal{T}_Y$ the preimage of V under f,

$$f^{-1}(V) = \{x \in X : f(x) \in V\}$$

is open in X, i.e. $f^{-1}(V) \in \mathcal{T}_X$.

If X and Y are metric spaces with the metric topologies, then the definition of a continuous map is equivalent the common $(\epsilon - \delta)$-definition of continuity from analysis, see [32, Theorem 21.1].

Definition 3.7. Let X and Y be topological spaces. A **homeomorphism** is a continuous bijection $f : X \to Y$ such that $f^{-1} : Y \to X$ is also a continuous bijection. If there exists a homeomorphism between X and Y we say that they are **homeomorphic** spaces.

Homeomorphisms formalises the idea of *continuous transformations* of topological spaces. The intuition is that if a space X can be continuously deformed into another space Y, then X and Y are, from a topological perspective, "the same". This is similar to how two isomorphic groups are "the same" from an algebraic perspective. From the definition of homeomorphisms one can easily deduce that homeomorphism define an equivalence relation on topological spaces. This is the equivalence relation we refer to when we are talking about "up to homeomorphism".

Since a homeomorphism gives a 1-1 correspondence between elements and open sets in X and Y, it preserves all properties that can be expressed in terms of the elements and topology of a space. Any such property is called a *topological property* or a *topological invariant*, since it is invariant under homeomorphisms. The aim of topology as a mathematical branch is to classify spaces up to homeomorphism. Considering spaces up to homeomorphism allows us to *disregard* properties of spaces that are not invariant under homeomorphisms.

3.2. **Homotopy Equivalence.**

Classifying spaces up to homeomorphism turns out to be incredibly hard. In many cases, we therefore classify spaces up to *homotopy equivalence* instead. This is a coarser equivalence relation, in the sense

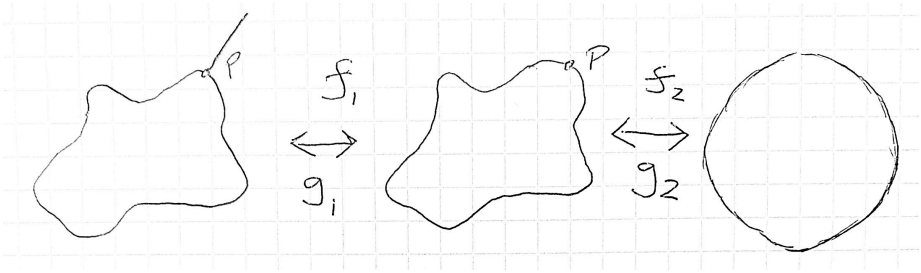


FIGURE 2. Three topological spaces and maps between them. The maps f_1 and g_1 are homotopy equivalences, but not homeomorphisms. The maps f_2 and g_2 are homeomorphisms.

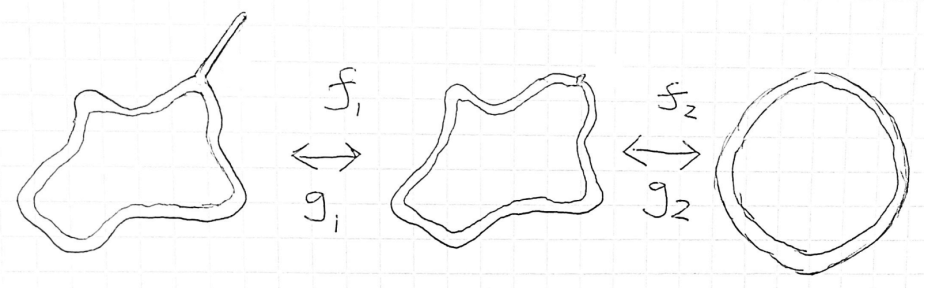


FIGURE 3. Three topological spaces and maps between them. All maps are homeomorphisms. The spaces are homotopy equivalent, but not homeomorphic, to the circle S^1.

that any two spaces that are homeomorphic are also homotopy equivalent. On the other hand, many spaces are homotopy equivalent, but not homeomorphic. *Homology*, which is an important topological invariant in this thesis, is invariant under homotopy equivalences.

Homotopy equivalences comes from the definition of *homotopic maps*. Informally, two maps are homotopic if one can be continuously deformed into the other.

Definition 3.8. Let X and Y be topological spaces and $f, g : X \to Y$ continuous maps. If there exists a continuous map $H : I \times X \to Y$ satisfying

$$H(0, x) = f(x) \quad \text{and} \quad H(1, x) = g(x)$$

for all $x \in X$, we say that f and g are **homotopic** maps. We call H a **homotopy** between f and g.

Here $I \subseteq \mathbb{R}$ is the interval $[0, 1]$ with the metric topology and $I \times X$ has the product topology.

It follows fairly easily from the definition that homotopy defines an equivalence relation on the set of continuous functions between topological spaces X and Y. We write $[f]$ for the equivalence class of maps that are homotopic to f.

Definition 3.9. Let X and Y be topological spaces and $f : X \to Y$ a continuous map. If there exists a continuous map $g : Y \to X$ such that the composition $f \circ g$ is homotopic to the identity map id_Y on Y and the composition $g \circ f$ is homotopic

to the identity map id_X on X, we say that f is a **homotopy equivalence**. If such f and g exist, we say that X and Y are **homotopy equivalent** spaces.

This definition is very similar to Definition 3.7 of a homeomorphism. The definition of homeomorphism could be phrased in the same format as Definition 3.9, but requiring

$$f \circ g = \mathrm{id}_Y \quad \text{and} \quad g \circ f = \mathrm{id}_X,$$

where the homotopy equivalence definition requires

$$[f \circ g] = [\mathrm{id}_Y] \quad \text{and} \quad [g \circ f] = [\mathrm{id}_X].$$

Hence, homotopy equivalence is a weaker notion of similarity between spaces than homeomorphism.

Example 3.10. In Figure 2, the spaces are locally homeomorphic to the interval $\mathbb{R}$, except at the point P in the leftmost space. The maps f_1 and g_1 are examples of homotopy equivalences that are not homeomorphisms, since the flare can not be continuously deformed and becomes a part of the loop. This can be seen from the fact that the interval I is not homeomorphic to a point. In Figure 3, the spaces are all two-dimensional manifolds, that is they are locally homeomorphic to $\mathbb{R}^2$, so the flare can be continuously shortened until it is part of the deformed annulus, thus all maps in Figure 3 are homeomorphisms.

4. Simplicial Complexes

In topological data analysis, *simplicial complexes* are an essential ingredient, both for persistent homology and, to a lesser extent, for Mapper. We can represent many topological spaces as purely combinatorial *abstract simplicial complexes*. Starting with the point cloud P, we will construct simplicial complexes that should resemble the space of interest K. For more details on simplicial complexes and simplicial homology see the book on algebraic topology by Munkres [33]. The majority of this section is taken from "An Introduction to Persistent Homology" [40] written by me.

Definition 4.1. A set of points $\{p_0, p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$ is said to be **geometrically independent** if the vectors $(p_1 - p_0), (p_2 - p_0), \ldots, (p_n - p_0)$ are linearly independent.

Definition 4.2. Given a geometrically independent set of points $\{p_0, p_1, \ldots, p_n\}$, the n**-simplex** spanned by these points is the set

$$\left\{ \sum_{i=0}^{n} t_i p_i : t_i \geq 0, \sum_{i=0}^{n} t_i = 1 \right\},$$

that is the set of convex combinations of the points; the convex hull. The subset of the n-simplex opposite the j-th vertex, that is

$$\left\{ \sum_{i=0}^{n} t_i p_i : t_i \geq 0, \sum_{i=0}^{n} t_i = 1, t_j = 0 \right\},$$

is called the j-th **face** of the n-simplex.

We say that an n-simplex has dimension n, since it can be thought of as a subspace of $\mathbb{R}^n$.

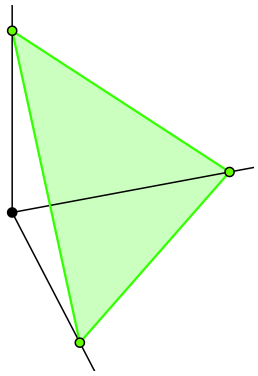


FIGURE 4. The standard 2-simplex. Illustration taken from [62].

Example 4.3. The 2-simplex spanned by the points $\{(1,0,0),(0,1,0),(0,0,1)\}$ in $\mathbb{R}^3$ is an equilateral triangle, as illustrated in Figure 4. This simplex is called the *standard 2-simplex*. In general, the n-simplex spanned by the unit vectors in $\mathbb{R}^{n+1}$ is called the *standard n-simplex*.

Example 4.4. Let Δ_n be the standard n-simplex and Δ'_n any n-simplex, both in $\mathbb{R}^{n+1}$. Define $f : \Delta'_n \to \Delta_n$ on $x \in \Delta'_n$ by

$$f(x) = \sum_{i=0}^{n} t_i e_i, \quad x = \sum_{i=0}^{n} t_i p_i$$

where e_i is the i-th unit vector in $\mathbb{R}^{n+1}$ and $\{p_i\}_i$ are the points spanning Δ'_n. Clearly, f is a homeomorpism. Thus, any n-simplex is homeomorphic to the standard n-simplex (and equivalently all n-simplices).

For spaces that are sufficiently well-behaved from a topological perspective we can create a homeomorphic (or homotopy equivalent) space built by simplices. The space built by simplices can often be easier to investigate.

Remark 4.5. We can never have an infinite amount of data, so the spaces we deal with in a data-setting will not be of a pathological sort.

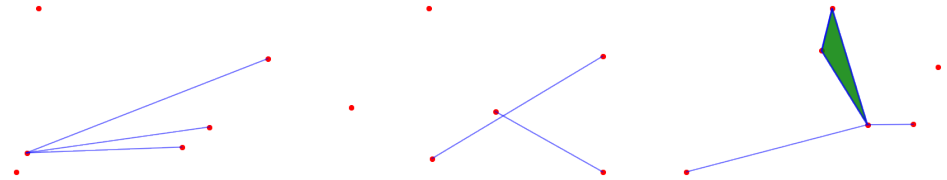
Definition 4.6. A **simplicial complex** L is a finite collection of simplices such that

- Every face of a simplex in L is also in L.
- If any two simplices in L have a non-empty intersection, then the intersection is also in L.

If L has a d-simplex, but no $(d+1)$-simplices, we say that L has dimension d. If K is a simplicial complex that consists only of simplices that are in L, then K is a sub-complex of L.

Since every face of a simplex in a simplicial complex L is also in L, we see in particular that the union of the vertices of all simplices in L are in L. These points are the 0-simplices and we call the set of these the **vertex set**.

Example 4.7. Let (V, E) be a graph with vertices V and edges E. Then (V, E) is a simplicial complex with vertex set V and the edges as 1-simplices spanned by the



(A) A graph and a simplicial complex embedded in $\mathbb{R}^2$

(B) A graph that is embedded such that it has an intersection

(C) A simplicial complex

FIGURE 5. Graphs and simplicial complexes.

two endpoints of each edge in E. We need to be careful with how we embed the vertex set so that no edges intersect, except for in the vertices. This is illustrated in Figure 5. The graph in Figure 5b can also be associated with a simplicial complex, but then we need to embed the points in a suitable way, potentially in a higher-dimensional space.

In this way, we can consider a simplicial complex as a higher-order generalisation of graphs, where we not only have edges between points, but also higher-order simplices. This is illustrated in Figure 5c. For the generalised graph to be a simplicial complex, we must require that all edges that enclose a filled-in triangle are edges in the graph, since all faces of a simplex in L must also lie in L. This is of course also necessary for higher-order generalisations.

As shown in Example 4.4, all n-simplices are homeomorphic. This means that we can capture the topology of a simplicial complex in the dimensions of the simplices and how they are connected. For this reason we introduce the purely combinatorial *abstract simplicial complex*.

Definition 4.8. An **abstract simplicial complex** A on the (vertex) set V, is a finite collection of non-empty subsets of V, such that if $\sigma \in A$, then all non-empty subsets of σ are also in A.

Example 4.9. Let L be a finite simplicial complex with vertex set $V = \{p_1, p_2, \dots, p_m\}$. Let the set $\{q_0, q_1, \dots, q_n\} \subseteq V$ lie in A, if and only if the n-simplex spanned by the points $\{q_0, q_1, \dots, q_n\}$ lies in L. Then A is an abstract simplicial complex.

Indeed, let ρ be $\{q_0, q_1, \dots, q_n\}$ with one point removed. The simplex spanned by this subset is a face of σ and so the simplex lies in L. Therefore, the set ρ lies in A. By induction, A satisfies definition 4.8.

If we let the elements in the sets of an abstract simplicial complex L correspond to points in $\mathbb{R}^d$, satisfying certain criteria, we can associate a simplicial complex $|L|$ with the abstract one. The simplicial complex $|L|$ is a *geometric realisation* of L. The abstract simplicial complex L is purely combinatorial, while $|L|$ is a topological space consisting of the union of all the simplices in L. An abstract simplicial complex with n vertices has a trivial geometric realisation in $\mathbb{R}^n$ by associating the vertices with the unit vectors. We could also associate the vertices with a geometrically independent set of points and consider the $(n-1)$-simplex spanned by these points. The simplicial complex is then a sub-complex of the simplicial

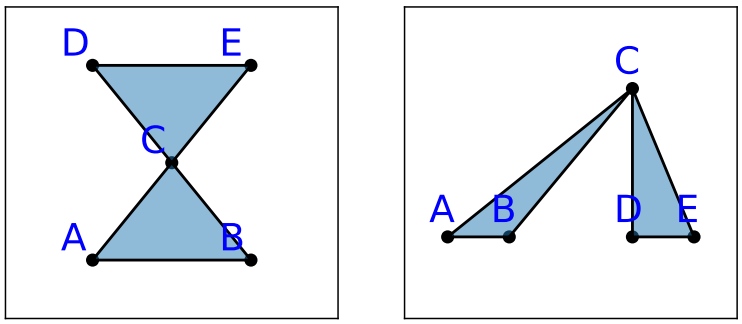


FIGURE 6. Geometric realisations of the abstract simplicial complex $\{A, B, C, D, E, AB, AC, BC, CD, CE, DE, ABC, CDE\}$. Associating different points with the vertices of the abstract simplicial complex results in different geometric realisations.

complex consisting of the $(n-1)$-simplex with all its faces, faces of the faces and so on. The simplicial complex can thus be realised as a subspace of the $(n-1)$-simplex which again can be considered a subspace of $\mathbb{R}^{n-1}$. It can also be shown that any abstract simplicial complex of dimension d has a geometric realisation in $\mathbb{R}^{2d+1}$. Conversely, any geometric simplicial complex has an associated abstract simplicial complex. The geometric simplicial complex is a geometric realisation of its associated abstract simplicial complex.

Example 4.10. In Figure 6, we see how an abstract simplicial complex can be geometrically realised as a simplicial complex. We associate the vertices in the abstract simplicial complex with points in a space, in this case $\mathbb{R}^2$. Different choices of points will result in different geometric realisations.

For the remainder of this thesis, "simplicial complexes" will be abstract simplicial complexes, and "realised simplicial complexes", will refer to the simplicial complex from Definition 4.6, which is an actual topological space.

5. Simplicial Homology

One of the most essential tools from algebraic topology is homology. For any topological space, we assign groups, the homology groups, that are invariant under homotopy equivalences (and therefore also homeomorphisms). This provides a good tool to distinguish topologically distinct spaces. To study the topology of data we will use persistent homology, a method that is based on computing the homology groups of spaces associated with the data at different "scales". We will here introduce simplicial homology briefly. For a thorough introduction to simplicial homology, see [33, Chapter 1].

Definition 5.1. Let L be a simplicial complex. A **simplicial n-chain on** L with coefficients in the abelian group G, is a finite sum of the form

$$\sum_{\lambda \in \Lambda} c_\lambda \sigma_\lambda,$$

where $\sigma_\lambda \in L$ is an n-simplex and $c_\lambda \in G$, for all λ in some finite index set Λ.

The group of simplicial n-chains on L over G is written $C_n(L;G)$. This is the free abelian group on the set of n-simplices in L. Addition is defined component-wise. If G is a field, then $C_n(L;G)$ is a vector space. We write $C_\bullet(L;G)$ to refer to the groups of simplicial n-chains in all dimensions. We have that $C_\bullet(L;G)$ is a chain complex, see [19, Chapter 2.1], but we will not bother to introduce chain complexes here.

Remark 5.2. Usually, we will have coefficients in $\mathbb{Z}, \mathbb{R}, \mathbb{Q}$ or in the finite fields $\mathbb{F}_p$. The finite field $\mathbb{F}_2$ creates a particularly simple situation where signs and orientations do not matter.

An n-simplex has $(n-1)$-simplices as its faces. These faces constitute the n-simplex' boundary.

Definition 5.3. We can define a map on an n-simplex $\sigma = \{p_0, p_1, \dots, p_n\}$ by

$$\partial(\sigma) = \sum_{i=0}^{n} (-1)^i \partial_i \sigma,$$

where $\partial_i \sigma$ is the i-th face of σ, that is the set $\{p_0, \dots p_{i-1}, p_{i+1}, \dots, p_n\}$. We can extend linearly to obtain a unique group homomorphism $\partial : C_n(L;G) \to C_{n-1}(L;G)$, that is, we define

$$\partial\left(\sum_{\lambda\in\Lambda} c_\lambda \sigma_\lambda\right) = \sum_{\lambda\in\Lambda} c_\lambda \partial(\sigma_\lambda),$$

which can easily be verified to be a group homomorphism. We call this operator the **boundary operator**.

Remark 5.4. Strictly speaking, this defines a boundary operator ∂ on $C_n(L;G)$ for each $n \in \mathbb{N}$, but we will understand which of these boundary operators we are dealing with from the dimension of the n-chains we apply it to.

Remark 5.5. The sign in Definition 5.3 captures the orientation of the simplices. Since the definition uses the ordering of the vertices that form a simplex, and we have defined abstract simplices as sets, we need to be a bit careful. Therefore, we require a total ordering of the vertex set, and denote simplices with the vertices in increasing order. In practice, we usually use coefficients in $\mathbb{F}_2$, so the signs in Definition 5.3 do not matter, and therefore the ordering of vertices is irrelevant.

A straightforward computation of $\partial(\partial(\sigma))$, for an n-simplex σ, will show that every face of a face occurs one time with positive sign and one time with a negative sign. This yields the following result, which is necessary for the construction of the homology groups.

Lemma 5.6. *The composition* $\partial \circ \partial = 0$.

Definition 5.7. The **group of simplicial n-cycles** $Z_n(L;G) \subseteq C_n(L;G)$ is the kernel of the boundary operator, i.e.

$$Z_n(L;G) = \ker(\partial : C_n(L;G) \to C_{n-1}(L;G)).$$

The **group of simplicial n-boundaries** $B_n(L;G) \subseteq C_n(L;G)$ is the image of the boundary operator, i.e.

$$B_n(L;G) = \mathrm{Im}(\partial : C_{n+1}(L;G) \to C_n(L;G)).$$

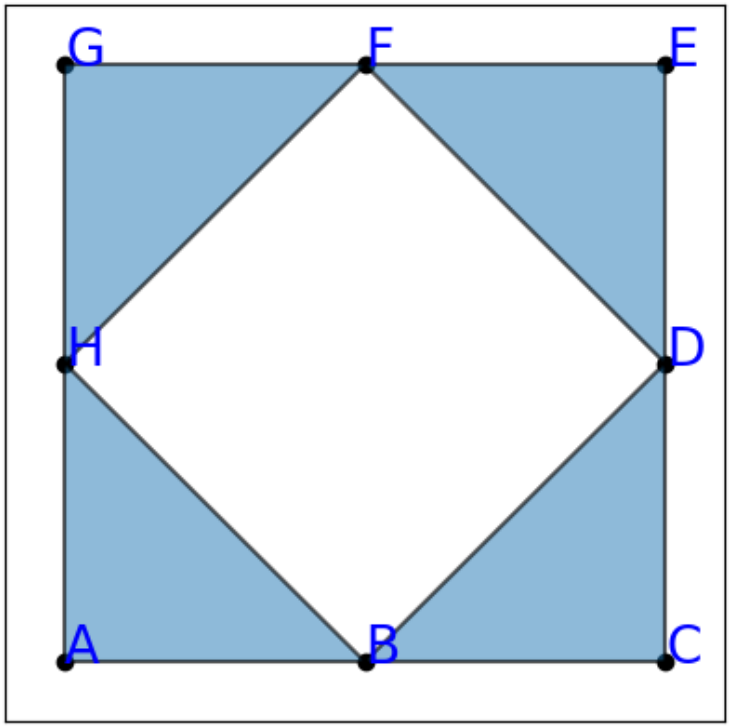


FIGURE 7. A simplicial complex L.

The n**-th simplicial homology group** $H_n(L;G)$ is the quotient group

$$H_n(L;G) = Z_n(L;G)/B_n(L;G).$$

Remark 5.8. The above defines simplicial homology of a simplicial complex. Based on similar ideas, we can define *singular homology* of any topological space, but this is beyond the scope of this thesis. The takeaway here is that (singular) homology is defined for any topological space, so it does for example make sense to speak about the homology of P_r. Moreover, if a space X is homotopy equivalent to the geometric realisation of a simplicial complex L, then the simplicial homology of L coincides with the singular homology of X. See the lecture notes by Haugseng [20] for a good introduction to singular homology, or Hatcher [19, Chapter 2.1] for definitions and proof of the above result.

The homology groups consists of the cycles that are not boundaries, and two cycles are considered the same if they differ by a boundary.

In Figure 7 we see a simplicial complex, say L. Let us investigate the 0-dimensional simplicial homology group $H_0(L;\mathbb{F}_p)$ of this simplicial complex. We order the simplices lexicographically to allow $p \neq 2$. We denote the n-simplices without the set brackets, e.g. $ABH = \{A, B, H\}$. The vertex A is clearly a 0-cycle. In fact, any vertex is a 0-cycle. In particular, B is also a 0-cycle. But we see that $B - A = \partial AB$, that is, A and B differ by a boundary, so $[A] = [B]$ in $H_0(L;\mathbb{F}_p)$. Similarly B and C differ by a boundary and belong to the same class in $H_0(L;\mathbb{F}_p)$.

We thus see that the group $H_0(L;\mathbb{F}_p)$ consists of one class for each path-component in L.

Let us now consider the 1-dimensional simplicial homology group $H_1(L;\mathbb{F}_p)$. Consider the simplicial 1-chain $BD + DF + FH - BH$. We see that in

$$\partial(BD + DF + FH - BH)$$

every vertex B, D, F and H occurs one time with a positive sign and once with a negative sign, so $BD + DF + FH - BH$ is a 1-cycle. The 1-chain $BC + CD - BD$ is also a 1-cycle, by the same argument, but this is exactly the boundary of the 2-simplex BCD, so $[BC + CD - BD] = 0$ in $H_1(L;\mathbb{F}_p)$. The 1-chain $BC + CD + DF + FH - BH$ is also a 1-cycle, but it differs from $BD + DF + FH - BH$ by the boundary of BCD, so they belong to the same homology class. After continuing this process, we would conclude that $H_1(L;\mathbb{F}_p)$ consists of 1 non-zero element, represented by $BD + DF + FH - BH$.

Notice that the simplicial complex in Figure 7 is homotopy equivalent to the circle S^1, and hence, they have the same homology groups in all dimensions, namely

$$H_0(S^1;\mathbb{F}_p) = \mathbb{F}_p, \quad H_1(S^1;\mathbb{F}_p) = \mathbb{F}_p$$

and $H_n(S^1;\mathbb{F}_p) = 0$ for all $n \geq 2$.

Intuitively, a 1-cycle is a 1-boundary if and only if everything the cycle encloses is also part of the simplicial complex. Moreover, two 1-cycles enclose the same hole if and only if they differ by a boundary. The higher-dimensional intuition is similar, but holes are replaced by n-dimensional voids. Intuitive interpretation can also become significantly more tricky when the homology groups involve torsion subgroups, as happens for example for non-orientable spaces like the Klein bottle or the real projective space $\mathbb{RP}^2$.

6. Persistent Homology

Recall the problem description: We have a metric space M, a point cloud $P \subseteq M$, sampled from the space $K \subseteq M$, and we want to study the topology of K. We need to associate topologically meaningful spaces to the point cloud P, and the union of closed balls

$$P_r = \{x \in M : \min_{p \in P} d(p, x) \leq r\}$$

is a good candidate for such a space. Then P_r is also a subspace of M and has the metric topology. We could represent the space P_r as a realised simplicial complex and compute the simplicial homology. This would give us information about the topology of P_r and under certain assumptions about K.

For now, assume r is fixed. We want to compute the simplicial homology of P_r. The *Čech complex* is a natural candidate for a simplicial complex that is closely related to the union of balls P_r.

Definition 6.1. Given a metric space M and a point cloud $P \subseteq M$, the **Čech complex** of P at scale r is

$$\check{\text{C}}\text{ech}_r(P) = \left\{ \sigma \subseteq P : \bigcap_{p \in \sigma} \overline{B_r(p)} \neq \emptyset \right\},$$

where $\overline{B_r(p)}$ denotes the closed ball of radius r, centred at p.

The Nerve Theorem establishes the relationship between P_r and $\check{\text{Cech}}_r(P)$. We first need to define *nerves*.

Definition 6.2. Let $\mathcal{U} = \{U_i\}_{i \in I}$ be a finite collection of sets. The **nerve** of $\mathcal{U}$ is the simplicial complex

$$\mathcal{N}(\mathcal{U}) = \{\sigma : \bigcap_{i \in \sigma} U_i \neq \emptyset\}.$$

Theorem 6.3 (The Nerve Theorem)**.** *Let $\mathcal{U}$ be a collection of closed convex sets in $\mathbb{R}^n$ such that any intersection between sets in $\mathcal{U}$ is contractible or empty. Then $|\mathcal{N}(\mathcal{U})|$ is homotopy equivalent to $\bigcup_{U \in \mathcal{U}} U$.*

This theorem is typically stated without proof in the literature. See for example the book by Edelsbrunner and Harer [14, p. III.2] for a general statement of the theorem. For a discussion and proof of another formulation of the theorem see Hatcher [19, Corollary 4G.3].

Remark 6.4. A *contractible space* is a space that is homotopy equivalent to a point.

The Nerve Theorem tells us that the geometric realisation of the Čech complex is actually homotopy equivalent to the union of balls P_r. The homology of the space P_r is therefore the same as the simplicial homology of the Čech complex of P at scale r. The simplicial homology of $\check{\text{Cech}}_r(P)$ can be computed through the reduction of a matrix representing the boundary operator in the basis given by the simplices in all dimensions in the Čech complex [64]. From the reduced matrix we can systematically read off the homology groups of the simplicial complex in all dimensions.

Example 6.5. Let us compute the simplicial homology of the Čech complex in Figure 8, with coefficients in $\mathbb{F}_p$. Denote the simplicial complex by X. The complex consists of the vertices

$$X_0 = \{A, B, C, D, E, F\},$$

the 1-simplices

$$X_1 = \{AB, AC, BC, BD, CF, DE, DF, EF\}$$

and the 2-simplices

$$X_2 = \{ABC, DEF\}.$$

Up to a maximal given dimension of interest n, the boundary operators in all dimensions form a linear transform on vector spaces

$$C_0(X; \mathbb{F}_p) \oplus \cdots \oplus C_n(X; \mathbb{F}_p) \to C_0(X; \mathbb{F}_p) \oplus \cdots \oplus C_{n-1}(X; \mathbb{F}_p).$$

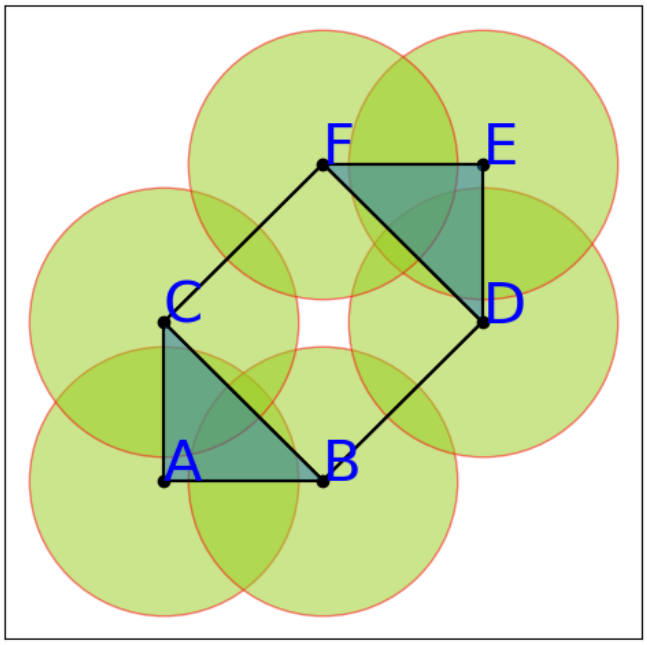


FIGURE 8. The Čech complex of the point cloud with 6 points A, B, C, D, E and F at the scale indicated by the green disks.

We can therefore represent the boundary operator as a linear transformation represented by the matrix

$$\mathcal{D} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ AB \\ AC \\ BC \\ BD \\ CF \\ DE \\ DF \\ EF \\ ABC \\ DEF \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} A & \dots & F & AB & AC & BC & BD & CF & DE & DF & EF & ABC & DEF \end{array} \\ \begin{pmatrix} & & & 1 & 1 & & & & & & & & \\ & & & -1 & & 1 & 1 & & & & & & \\ & & & & -1 & -1 & & 1 & & & & & \\ & & & & & & -1 & & 1 & 1 & & & \\ & & & & & & & & -1 & & 1 & & \\ & & & & & & & -1 & & -1 & -1 & & \\ & & & & & & & & & & & 1 & \\ & & & & & & & & & & & -1 & \\ & & & & & & & & & & & 1 & \\ & & & & & & & & & & & & \\ & & & & & & & & & & & & 1 \\ & & & & & & & & & & & & -1 \\ & & & & & & & & & & & & 1 \\ & & & & & & & & & & & & \\ & & & & & & & & & & & & \end{pmatrix} \end{array},$$

where all the other entries are 0. Simplicial homology can now be computed by reducing the boundary operator matrix. In our case, this results in

$$\dim H_0(X;\mathbb{F}_p) = 1,$$

since any pair of vertices differ by the boundary of a path between them. We also get

$$\dim H_1(X;\mathbb{F}_p) = 1,$$

since the cycle $BD+DF-CF-BC$ is not a boundary. The cycles $AB+BC-AC$ and $DE+EF-DF$ are both boundaries, while the cycles like $AB+BD+DF-FC-AC$ differ from $BD+DF-CF-BC$ by a boundary. The 2-dimensional homology group $H_2(X;\mathbb{F}_p)$ is zero.

At this point we can make two observations that solve the question of how to choose the radius r. We will work with coefficients in a finite field $\mathbb{F}_p$, so the homology groups will be vector spaces.

Observation 6.6. For $r' \leq r$ we have that $\check{\text{C}}\text{ech}_{r'}(P) \subseteq \check{\text{C}}\text{ech}_r(P)$, similarly to how we have $P_{r'} \subseteq P_r$. For a finite point cloud P, we have a finite number of values of r where $\check{\text{C}}\text{ech}_r(P)$ changes. In other words, there exist $r_0, r_1, \ldots, r_n$ such that

$$\check{\text{C}}\text{ech}_{r_0}(P) \subseteq \check{\text{C}}\text{ech}_{r_1}(P) \subseteq \cdots \subseteq \check{\text{C}}\text{ech}_{r_n}(P), \tag{1}$$

and for any $r \in \mathbb{R}$, we have that $\check{\text{C}}\text{ech}_r(P) = \check{\text{C}}\text{ech}_{r_i}(P)$, for some $0 \leq i \leq n$.

Observation 6.7. Define the **scale of a n-simplex** σ to be the smallest r_i such that $\sigma \in \check{\text{C}}\text{ech}_{r_i}(P)$ from (1). Let $\mathcal{D}$ be the boundary operator on $C_\bullet(\check{\text{C}}\text{ech}_{r_n}(P);\mathbb{F}_p)$, represented in the basis consisting of all the simplices in $\check{\text{C}}\text{ech}_{r_n}(P)$, ordered in increasing scale firstly and increasing dimension secondly. Then $\mathcal{D}$ is upper diagonal, since the boundary of a simplex has lower dimension and is a sum of simplices with smaller or equal scale. It follows that if we reduce the matrix D, we have also reduced the sub-matrices that represent the boundary operators on $C_\bullet(\check{\text{C}}\text{ech}_{r_i}(P);\mathbb{F}_p)$ for all $0 \leq i \leq n$.

Example 6.8. Consider again the simplicial complex in Figure 8. The basis for the boundary operator matrix is in this example already ordered as described in Observation 6.7. Let Y be the smaller Čech complex that consists of the same vertices and edges, but without the 2-simplices. Let $\mathcal{D}'$ be the boundary operator matrix for Y. Then $\mathcal{D}'$ is the submatrix of $\mathcal{D}$ obtained by removing the two rows and columns corresponding to ABC and DEF. If we reduce $\mathcal{D}$ with a method that respects the ordering of the basis, then $\mathcal{D}'$ reduced is a submatrix of $\mathcal{D}$ reduced.

If Z is the Čech complex from Figure 8, but also containing the edges BF and CD and the 2-simplices BCD, BCF, BDF and CDF, the ordering of the basis should be

$$A, B, \ldots, F, AB, AC, \ldots, EF, ABC, DEF, BF, CD, BCD, BCF, BDF, CDF.$$

By computing the reduced boundary operator matrix with this basis we get the reduced versions of both $\mathcal{D}$ and $\mathcal{D}'$ as submatrices.

We see that if we compute the simplicial homology of $\check{\text{C}}\text{ech}_r(P)$, we simultaneously compute the simplicial homology of $\check{\text{C}}\text{ech}_{r'}(P)$ for all $r' \leq r$, and there are a finite number of relevant such values r'.

We can actually retrieve more information from the reduced version of $\mathcal{D}$. By starting with only the simplices of scale r_0 we can find generators (which must be

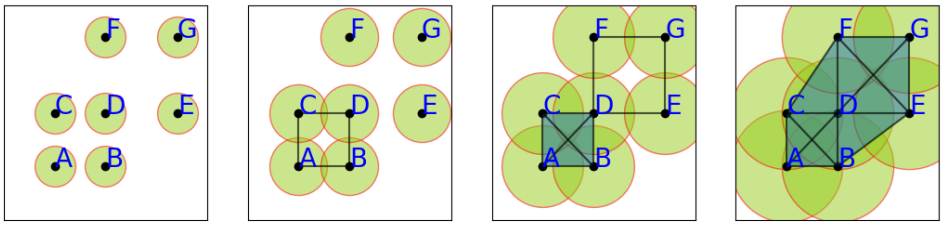


FIGURE 9. Čech complexes at different scales, indicated by the green disks.

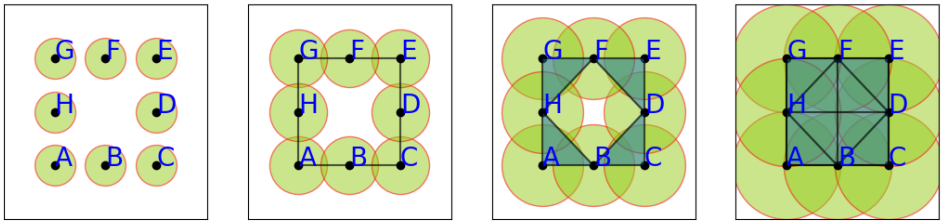


FIGURE 10. Čech complexes at different scales, indicated by the green disks.

cycles that are not boundaries) for the homology groups of different dimensions. When we include the simplices of scale r_1, some of the generators may become boundaries, some new cycles may appear and some of the generators will remain generators. We can continue this process and include the simplices of scale r_2, r_3 up to r_n. Subject to being able to represent the generators in a suitable basis, we can track the *birth* and *death* of the generators of different dimensions as we increase the scale.

Not only have we computed the homology of P_r at all values of r, but we also track the generators across different scales. We could for example identify that a particular loop is present in P_{r_i}, P_{r_j} and all values of r between r_i and r_j. That is far more useful than being able to say that we know there is a loop in both P_{r_i} and P_{r_j}, but without knowing whether it is the same loop or not. This is the idea behind *persistence*.

Remark 6.9. We can always represent the generators in a suitable basis, see the paper on computing persistent homology [64]. The result is, however, best explained using functoriality and persistence modules, which we will not discuss here. See [36, Chapter 1.2] or [40, Section 7.2] for the result.

Example 6.10. Consider the Čech complexes at different scales in Figure 9 and Figure 10. Let $X^{(1)}, X^{(2)}, X^{(3)}$ and $X^{(4)}$ denote the simplicial complexes from left to right in Figure 9 and $Y^{(1)}, Y^{(2)}, Y^{(3)}$ and $Y^{(4)}$ the complexes in Figure 10. For ease of notation, let us define the n-th Betti number of a simplicial complex Z with coefficients $\mathbb{F}_p$

$$\beta_n(Z;\mathbb{F}_p) = \operatorname{rank} H_n(Z;\mathbb{F}_p) = \dim H_n(Z;\mathbb{F}_p),$$

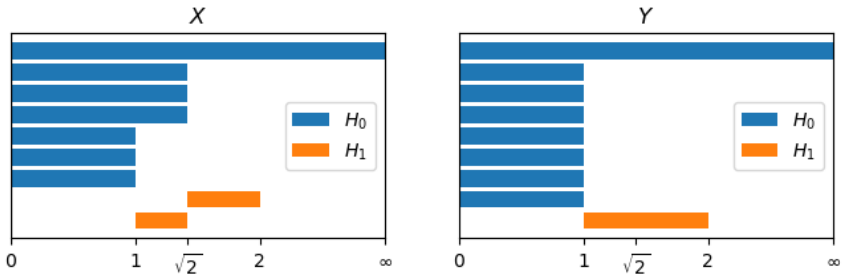


FIGURE 11. Persistence barcode for the simplicial complexes in Figure 9 (left) and Figure 10 (right). The x-axis is the scale of the Čech complexes of the point clouds in Figure 9 and Figure 10.

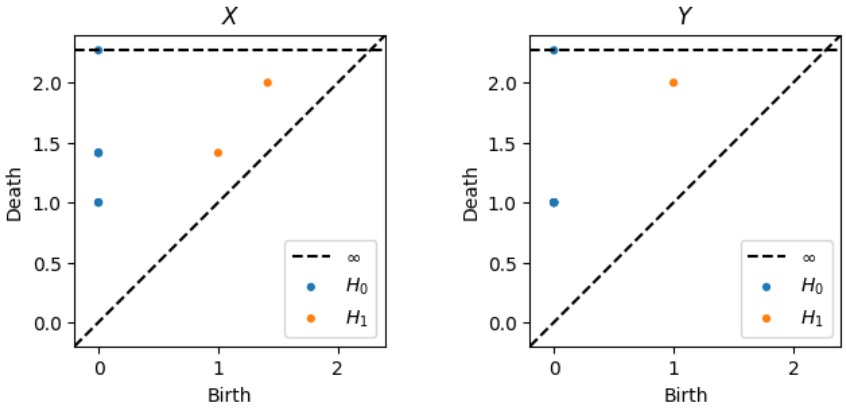


FIGURE 12. Persistence diagrams for the simplicial complexes in Figure 9 and Figure 10. The axes correspond to the scale of the Čech complexes.

where the last equality holds because $\mathbb{F}_p$ is a field and hence $H_n(Z;\mathbb{F}_p)$ is a vector space. We can compute the Betti numbers of the simplicial complexes and get that

$$\beta_0(X^{(1)}) = 7 \quad \beta_0(X^{(2)}) = 4 \quad \beta_0(X^{(3)}) = 1 \quad \beta_0(X^{(4)}) = 1$$
$$\beta_1(X^{(1)}) = 0 \quad \beta_1(X^{(2)}) = 1 \quad \beta_1(X^{(3)}) = 1 \quad \beta_1(X^{(4)}) = 0,$$

with coefficients in $\mathbb{F}_p$. This computation can be done efficiently by reducing only the boundary operator matrix for $X^{(4)}$. We also get

$$\beta_0(Y^{(1)}) = 8 \quad \beta_0(Y^{(2)}) = 1 \quad \beta_0(Y^{(3)}) = 1 \quad \beta_0(Y^{(4)}) = 1$$
$$\beta_1(Y^{(1)}) = 0 \quad \beta_1(Y^{(2)}) = 1 \quad \beta_1(Y^{(3)}) = 1 \quad \beta_1(Y^{(4)}) = 0,$$

The 0-dimensional Betti numbers are not the same but the 1-dimensional Betti numbers are the same for $X^{(i)}$ and $Y^{(i)}$ for all $1 \leq i \leq 4$. But the spaces we are investigating are quite different in terms of 1-dimensional holes. In Figure 9 there are two structures that result in non-zero Betti numbers, while in Figure 10 there is only 1, but this one seems more "significant", in some sense of the word. Let us

try to track the generators for the 1-dimensional homology groups in the complexes from left to right.

In $X^{(1)}$ there are no 1-cycles. In $X^{(2)}$ the 1-cycle $AB+BD-CD-AC$ is *born*. In $X^{(3)}$ the cycle $DE+EG-FG-DF$ is born, while the 1-cycle $AB+BD-CD-AC$ just *died*. In $X^{(4)}$ all 1-cycles are dead. To summarise, one generator is born in $X^{(1)}$ and dies in $X^{(2)}$, while one generator is born in $X^{(2)}$ and dies in $X^{(3)}$.

In Figure 10 a 1-cycle is born in $Y^{(2)}$, is present in $Y^{(3)}$ and dies in $Y^{(4)}$.

We can draw the birth and death of the generators as a *barplot*. This is done in Figure 11, where we have set the distances between the closest points in Figure 9 and Figure 10 to be 1. The x-axis in the plot corresponds to the scale of the Čech complex.

The barcode tells us the scale of birth and death of the generators of the different homology groups. Barplots are a common way to visualise the output from persistence homology. The most common way to visualise the output is as a *persistence diagram*. In a persistence diagram we represent each bar as a point with its birth and death as its coordinates. Figure 12 shows persistence diagrams of the barcodes from Figure 11. We can see the length of the bar, or the *persistence of the generator*, as the corresponding point's distance from the diagonal in the persistence diagram. In this example we see that the loop in Figure 10 is more *persistent* than those in Figure 9, and this seems to be because it is a more prominent feature of the point cloud.

The theory behind persistent homology justifies the idea that the most persistent features (long bars) correspond to global features in the data [14], while short-lived features (short bars) are attributed to noise, local features or local structure in the data [2]. In this thesis we will use persistent homology to detect global features in the data.

6.1. **The Vietoris–Rips complex.** When computing persistent homology (or cohomology) we rarely deal with the Čech complex, but rather the *Vietoris–Rips complex*. This is because the Vietoris–Rips complex is combinatorially simpler and better suited for computations.

Definition 6.11. The **Vietoris–Rips complex** at scale r of a point cloud P, in a metric space, is the simplicial complex

$$\mathrm{VR}_r(P) = \{\sigma \subseteq P : \mathrm{diam}(\sigma) \leq 2r\}.$$

The diameter of a subset σ is defined as

$$\mathrm{diam}(\sigma) = \max_{p,q\in\sigma} d(p,q),$$

where d is the metric of the metric space.

Note that the properties of the Čech complex that we use in Observations 6.6 and 6.7 also apply to the Vietoris–Rips complex, so we can replace the Čech complex with the Vietoris–Rips complex in the computation of persistent homology, of course with potentially different results.

Remark 6.12. From the definition, we can observe that a simplex σ is in $\mathrm{VR}_r(P)$ if and only if for any pair of vertices $p, q \in \sigma$ the edge $\{p, q\}$ is in $\mathrm{VR}_r(P)$. Thus, $\mathrm{VR}_r(P)$ is determined entirely by its vertices and edges. When computing persistent homology with Vietoris–Rips complexes this results in far fewer short-lived

homology generators than with the Čech complex. For example, when using the Čech complex, any three points forming an acute triangle will result in a short-lived generator in dimension 1. The number of scales of interests, as defined in Observation 6.6, will also be strictly smaller when we use the Vietoris–Rips complex.

The geometric realisation of the Vietoris–Rips complex is not homotopy equivalent to the union of balls P_r, but we can easily derive the relation

$$\check{\mathrm{C}}\mathrm{ech}_r(P) \subseteq \mathrm{VR}_r(P) \subseteq \check{\mathrm{C}}\mathrm{ech}_{2r}(P).$$

We can strengthen this relationship and obtain

$$\check{\mathrm{C}}\mathrm{ech}_r(P) \subseteq \mathrm{VR}_r(P) \subseteq \check{\mathrm{C}}\mathrm{ech}_{2\delta r}(P),$$

where $\delta = \sqrt{\frac{d}{2(d+1)}}$, see [14, Chapter III.2] for proof (they prove the above result, even though their statement of the result is weaker). Intuitively, we see that the Čech complex and the Vietoris–Rips complex are very similar. Using *functoriality* of homology, *interleaving distances* and the *stability* of persistent homology, we can prove that replacing the Čech complex with the Vietoris–Rips complex results in bounded small perturbation of the persistence diagram, see [36, Chapter 5.1].

All computations of persistent (co)-homology in this thesis are done with the Ripser software [4], which uses Vietoris–Rips complexes.

6.2. **Preprocessing for Persistent Homology.** Persistent homology is highly sensitive to outliers. For example, a single point in the middle of a circle will dramatically decrease the death time of the generator corresponding to that circle. Persistent homology is also impractical to compute for very large datasets ($>\sim$ 10000). In order to address these weaknesses, one usually does some preprocessing of the data before computing persistent homology. We do combinations of two kinds of preprocessing in this thesis: *Codensity Filtration* and *Denoising*.

6.2.1. *Codensity Filtration.*

Definition 6.13. Let P be a point cloud with a metric d. For a positive natural number k we define the **k-codensity** of a point $p \in P$ to be $d(p, y)$, where y is the k-th closest point to p in P.

Remark 6.14. The inverse $\frac{1}{d(p,y)}$ is an estimate for the density at point p. For small k this density approximation is quite local, while for large k the codensity essentially becomes an eccentricity measure, that is a measure of how far from the centre of data the point p lies.

We will use codensity to estimate the density of the point cloud P and then filter away points in low-density regions. This will reduce the number of points and focus on the shape of high density regions. This is in line with Remark 2.1 about considering the points in P to be samples drawn from a probability distribution on the metric space M. In this case we are interested in the topology of the subspace of high probability, which should correspond to regions of M that are densely sampled.

Figure 13 illustrates how a codensity filtration might help reveal underlying structure in noisy data. The codensity parameter k is essential to separate what is global and local structure of the data. The number of points we keep is also very essential to choose right, as illustrated by the figure.

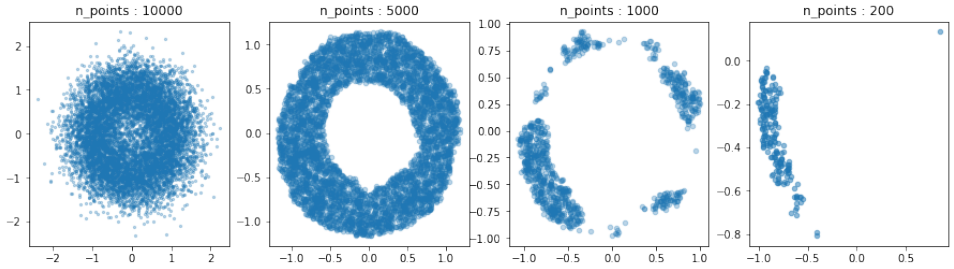


FIGURE 13. A circle with radius 1 sampled with Gaussian noise at different codensity filtrations levels. Codensity parameter $k = 500$ is the same for all plots.

6.2.2. *Denoising.* The spaces we work in are typically high-dimensional and therefore quite sparsely populated with points. Moving the points to lower-dimensional parts of the whole space is helpful in remedying the *curse of dimensionality* [57] for this application. We do this by an iterative denoising.

Definition 6.15 (Denoising)**.** Given a natural number k, the **denoising algorithm** can be described as follows.

$P' = \emptyset$
for p in P **do**
    add the centroid of p and its k nearest neighbours to P'
end for
return P'

Remark 6.16. If $P \subseteq M$ with $M \neq \mathbb{R}^m$ for some m, one might have to define what "centroid" of a set of points in M should mean, but we always have $M = \mathbb{R}^m$ in this thesis when we apply denoising.

Remark 6.17. Definition 6.15 refers to *one* iteration of the denoising algorithm. Some times we will repeat the denoising process a couple of times.

This denoising process has the property that it collapses high-dimensional subspaces spanned by the data points into lower-dimensional spaces. Figure 14 illustrates this on a dataset from a noisy circle. It is quite clear that the denoising process destroys the local structure of the dataset by introducing spurious local patterns, like those visible after 3 iterations. However, the global structure is preserved, at least after just a few iterations. The denoising parameter k from Definition 6.15 determines what is considered global structure. In the limit, the data will converge to a number of dense 0-dimensional clusters, as it is starting to do after 5 iterations in Figure 14. In practice, 1 or 2 iterations is typically chosen.

The denoising process can easily create structure that is not actually present in the data, as seen in the third plot in Figure 14. We illustrate this by doing denoising on random uniform data in Figure 15. After one iteration we see that the denoising shrinks the boundaries of the data, and therefore gets a high density region on the boundary. This results in spurious loop structures emerging after 3 and 5 iterations. We should therefore be a bit careful with results obtained after doing denoising. In addition, it seems to be a good idea to not do more than 1 iteration of denoising.

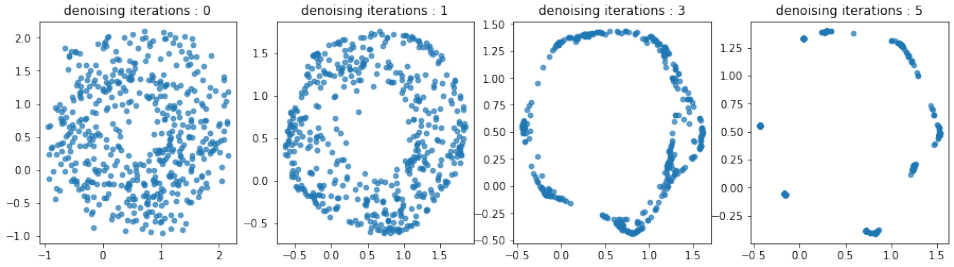


FIGURE 14. Different iterations denoising of a noisy circle. Denoising parameter $k = 50$. Uniform noise

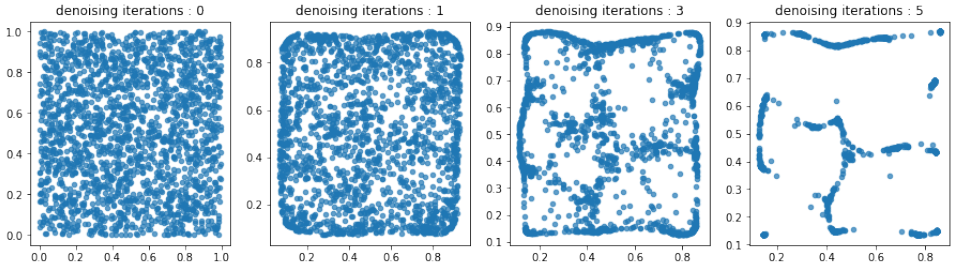


FIGURE 15. Different iterations denoising of random uniform data. Denoising parameter $k = 100$. Spurious structure arises when doing multiple iterations.

There is little reason to expect such kind of spurious structures arising after doing codensity filtration only.

7. Circular Parametrisation

Many times after identifying a persistent loop in a dataset, we might be interested in getting a parametrisation of the loop. This could be useful in order to investigate the loop further or to reduce the dimensionality of the dataset. *Simplicial cohomology*, the dual of simplicial homology, will give us a way to create such a parametrisation, following the work of de Silva, Morozov and Vejdemo-Johansson [47].

7.1. Simplicial Cohomology.

First we need to define simplicial cohomology.

Definition 7.1. Let L be a simplicial complex. An **n-cochain** in L over the group of coefficients G is a homomorphism $\phi : C_n(L;\mathbb{Z}) \to G$. The **group of n-cochains** in L over G is the group of such homomorphisms

$$C^n(L;G) = \mathrm{Hom}(C_n(L;\mathbb{Z}), G),$$

where addition is defined by $(\alpha+\beta)(\sigma) = \alpha(\sigma)+\beta(\sigma)$ for $\alpha, \beta \in \mathrm{Hom}(C_n(L;\mathbb{Z}), G)$ and $\sigma \in C_n(L;\mathbb{Z})$.

Remark 7.2. With coefficients in a field $\mathbb{F}$, the simplicial cohomology groups $C^\bullet(L;\mathbb{F})$ are vector spaces; the dual vector spaces of the simplicial homology groups $C_\bullet(L;\mathbb{F})$.

In order to define simplicial cohomology we need a coboundary operator.

Definition 7.3. The **coboundary operator** $\partial^n : C^n(L;G) \to C^{n+1}(L;G)$ is the group homomorphism defined by

$$\partial^n(\alpha)(\sigma) = \alpha(\partial_{n+1}(\sigma)),$$

for $\sigma \in C_{n+1}(L;\mathbb{Z})$ and $\alpha \in C^n(L;G)$, where $\partial_{n+1} : C_{n+1}(L;\mathbb{Z}) \to C_n(L;\mathbb{Z})$ is the boundary operator on simplicial $(n+1)$-chains.

Remark 7.4. The coboundary operator takes n-cochains to $(n+1)$-cochains, as opposed to the boundary operator which takes n-chains to $(n-1)$-chains.

We can define simplicial cohomology similarly to how we defined simplicial homology earlier.

Definition 7.5. Let L be a simplicial complex and G the group of coefficients. The **group of simplicial n-cocycles** $Z^n(L;G) \subseteq C^n(L;G)$ is the kernel of the coboundary operator, i.e.

$$Z^n(L;G) = \ker(\partial^n : C^n(L;G) \to C^{n+1}(L;G)).$$

The **group of simplicial n-coboundaries** $B^n(L;G) \subseteq C^n(L;G)$ is the image of the coboundary operator, i.e.

$$B^n(L;G) = \mathrm{Im}(\partial^{n-1} : C^{n-1}(L;G) \to C^n(L;G)).$$

The **n-th simplicial cohomology group** $H^n(L;G)$ is the quotient group

$$H^n(L;G) = Z^n(L;G)/B^n(L;G).$$

From the universal coefficient theorem for cohomology, see [19, Theorem 3.2], we can deduce that since $H_n(L;\mathbb{F})$ is a vector space for all n, that

$$H^n(L;\mathbb{F}_p) \cong H_n(L;\mathbb{F}_p).$$

In other words, simplicial homology and cohomology agree when we have coefficients in a field.

So what was the point of introducing simplicial cohomology when it turns out to be the same as simplicial homology in the case we are interested in? There are at least two answers to this question.

Notice that we can represent the coboundary operator as a matrix $\mathcal{D}'$ with respect to the same basis as in Observation 6.7, ordered in increasing scale firstly, but decreasing dimension secondly. This will again make the matrix $\mathcal{D}'$ upper diagonal and all the arguments following the observations the same. We can therefore compute persistent homology indirectly by computing persistent cohomology, and it turns out that this is considerably faster in practice (when combined with certain tricks) [4].

The main reason to introduce simplicial cohomology in this thesis is that it allows us to create a circular parametrisation of the loops we find with persistent (co)homology.

7.2. **Circular Parametrisation.** In the paper [47], de Silva, Morozov and Vejdemo-Johansson introduce a procedure for obtaining a *circular parameter* from the results of a computation of persistent cohomology. We start by making explicit what we mean by a circular parameter.

The dimensionality reduction problem is usually stated as follows: Given a point cloud P with a metric d, find a mapping $P \to \mathbb{R}^n$, for some, typically small, $n \in \mathbb{N}$,

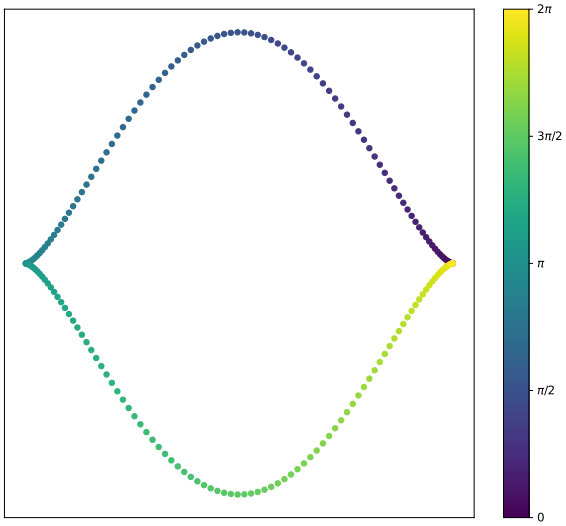


FIGURE 16. Data P sampled from a manifold $M \cong S^1$, coloured by the angle in S^1, under a homeomorphism $M \to S^1$.

preserving the intrinsic structure of the data. What is meant by "preserving the intrinsic structure of the data" could be defined in a variety of ways. For example, some methods aim to create an embedding that makes the smallest possible change to the pairwise distances between points, some do this but focusing only on the pairwise distances between nearby points (local methods) and others by emphasising the longer distances (global methods). Other methods assume that the data P is sampled from an n-dimensional manifold M, and should therefore be embeddable in $\mathbb{R}^n$ (at least locally), and aim to essentially find a homeomorphism between the manifold and the embedding space $\mathbb{R}^n$.

Common for most dimensionality reduction methods is that they aim to embed the data in the space $\mathbb{R}^n$, but this is clearly not a suitable space for all kinds of data. The simplest example is of course the circle S^1, which is a 1-dimensional manifold that cannot be embedded in $\mathbb{R}^1$. If we have data P that lies on a circle, or any loop that is homeomorphic to S^1, the best representation of the data is the distance along the loop from some point of reference. This is illustrated in Figure 16. We understand that the suitable dimensionality reduction for P is a function $\theta : P \to S^1$, not a function $P \to \mathbb{R}^n$ for some n. We call θ a *circular parametrisation* of P and the values associated with each point in P under θ, the *circular parameters*.

Finding a circular parametrisation of a loop in a dataset is considerably more difficult than it first might seem. This is in part due to the fact that data rarely looks like that in Figure 16. For example, the data might be of a higher dimension, it might be noisier so the loop is thicker, perhaps the loop does not constitute the whole space or the density of the data might change significantly across the loop. Moreover, there might be more than 1 loop present in the data, for example if the data lies on the torus $S^1 \times S^1$.

In the paper [47] by de Silva, Morozov and Vejdemo-Johansson, the authors design a circular parametrisation based on the equation from homotopy theory

$$[X, S^1] = H^1(X; \mathbb{Z}), \tag{2}$$

where X is a topological space with the homotopy type of a cell complex (see [33, Chapter 4 §38], and $[X, S^1]$ is the set (which also has a group structure that makes (2) a isomorphism [19, Theorem 4.57]) of homotopy equivalence classes of continuous maps $X \to S^1$. Since maps $X \to S^1$ are exactly the circular parametrisations we are looking for, non-zero elements in $H^1(X; \mathbb{Z})$, i.e. cocycles in X, provide us with a good starting point for creating circular parametrisations on X. If $H^1(X; \mathbb{Z}) = 0$, no loops are present in the data so we need not bother with looking for circular parametrisations.

As in the previous sections about persistent homology, we first encounter the problem that X in (2) is a topological space while the point cloud P we are dealing with has the discrete topology. We will again overcome this by associating a simplicial complex to the point cloud, typically a Vietoris–Rips complex $\mathrm{VR}_r(P)$. Through a computation of persistent cohomology, we find a significant cocycle to use as a starting point and identify a scale r of the simplicial complex where this cocycle is present. If we find no significant cocycles with persistent cohomology, we conclude that the data is unsuitable for a circular parametrisation. If we find a significant cocycle, we can choose as scale r some value between the scales of birth and death for this cocycle. Then the cocycle will be present in the simplicial complex $\mathrm{VR}_r(P)$, and we can use this cocycle to construct a circular parametrisation of the space $\mathrm{VR}_r(P)$.

A cocycle in $\mathrm{VR}_r(P)$ found with persistent cohomology is an element in $H^1(\mathrm{VR}_r(P); \mathbb{F}_p)$, not in $H^1(\mathrm{VR}_r(P); \mathbb{Z})$. However, through a few theoretically justified steps, see original paper [47], we can obtain a circular parametrisation $\theta : \mathrm{VR}_r(P) \to S^1$. In practice, the method takes as inputs a cocycle and the edges in the simplicial complex at the chosen scale, solves an optimisation problem that essentially minimises the change of the circular parameter along the edges of the simplicial complex, and returns the circular parameter for all vertices that are present in the cocycle. From the circular parameter values on the vertices, the circular parametrisation $\theta : \mathrm{VR}_r(P) \to S^1$ can be determined on the whole simplicial complex. This allows us to compute a circular parameter value for previously unseen points through interpolation, as long as they lie within the simplicial complex $\mathrm{VR}_r(P)$.

Constructing a circular parametrisation from significant cocycles (loops) detected with persistent cohomology makes it easier to unwrap the meaning of the loop, as well as providing us with a tool to indicate the loop in other visualisations of the data. Thus, the circular parametrisation procedure allows us to build stronger confidence in the results of a computation of persistent cohomology. Moreover, the quantitative nature of the circular parametrisation makes it useful for further quantitative investigations, as opposed to the qualitative nature of a result like "presence of significant cocycle(s)".

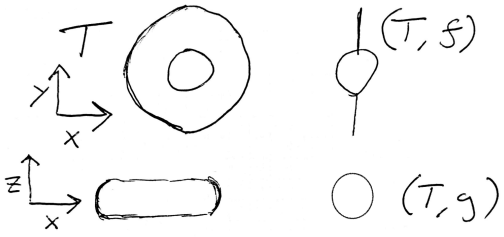


FIGURE 17. Reeb graphs of the torus for projections on different axes.

8. Mapper

In this section I will introduce Mapper; a tool for unsupervised exploratory analysis of topological properties of data, particularly useful for high-dimensional datasets. Mapper was first introduced in [50] by Singh et al.

Before we introduce Mapper, let us define Reeb graphs. As we will see later, Mapper can be understood as a discrete approximation of a Reeb graph.

Definition 8.1. Let X be a topological space and $f : X \to \mathbb{R}$ a continuous function on X. We will often refer to f as the **filter function**. For $t \in \mathbb{R}$, let $f^{-1}(t)$ be the preimage image of t under f in X, also known as the fibre. The Reeb graph of X with respect to f is the quotient space $X/\sim$, where $\sim$ is the equivalence relation such that $x \sim y$ if and only if $f(x) = f(y)$ and x and y lie in the same path-component in the fibre $f^{-1}(f(x))$.

Example 8.2. Let $\mathbb{T}$ be the surface of revolution that is homeomorphic to the two-torus $\mathbb{T}^2 = S^1 \times S^1$, embedded in $\mathbb{R}^3$ so that the central circle lies in the xy-plane. Let $f : \mathbb{T} \to \mathbb{R}$ be the projection on the y-axis and $g : \mathbb{T} \to \mathbb{R}$ the projection on the z-axis. In Figure 17 we see the Reeb graphs for the different functions. This illustrates the idea that the Reeb graph gives us a simplified version of the space X as seen through the lens of the filter function. Different functions can result in different Reeb graphs. The projection on the y-axis recovers the hole in the middle of the torus, while the projection on the z-axis recovers the enclosed hole of the torus.

We can also define a "Reeb graph", which will not necessarily be a graph, for filter functions that are $f : X \to \mathbb{R}^m$ for $m > 1$ in the same way as in Definition 8.1.

Let $f : X \to \mathbb{R}^2$ be the projection on any two non-parallel axes for the surface of revolution $\mathbb{T}$. The preimage under f of any point in $\mathbb{R}^2$ will then consist of the intersection between $\mathbb{T}$ and a straight line. This preimage will then consist of 0, 1, 2, 3 or 4 disconnected points. Thus the "Reeb graph" of $\mathbb{T}$ with respect to f is then homeomorphic to $\mathbb{T}$ itself.

Let $\mathbb{T}'$ denote the surface of revolution that is homeomorphic to the filled-in two-torus $S^1 \times D^2$, where D^2 is the unit disk. Again we can consider $\mathbb{T}'$ embedded as in Figure 17. The "Reeb graph" of $\mathbb{T}'$ with respect to the projection on the xy-plane is homeomorphic to the annulus, e.g. $\{x \in \mathbb{R}^2 : 1 \leq \|x\| \leq 2\}$. The "Reeb graph" of $\mathbb{T}'$ with respect to the projection on the xz-plane is homeomorphic to the upper right Reeb graph in Figure 17 times the unit interval I.

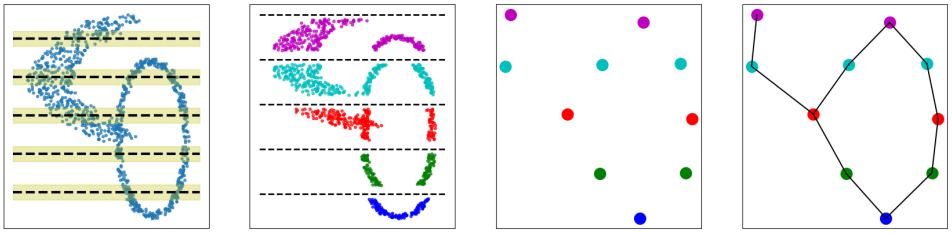


FIGURE 18. The Mapper pipeline. Left: The two dimensional data. The covering is illustrated with the lines and the overlap of the covers with the yellow region. The filter function is the projection on the vertical axis. Second from the left: The preimages of the covering. Second from the right: One node per cluster in each of the preimages. Right: We get a graph by adding edges between any clusters with points in common.

In our setting, we view data as a point cloud. In practice, P is not an interesting topological space, since it consists only of disconnected points. Any point in P is its own path-component. and therefore the Reeb graph of P is homeomorphic to P, regardless of the filter function. It is clear that in order to construct a meaningful analogy of the Reeb graph in the finite, discrete setting we are dealing with, we have to relax our notions of continuity.

The first way we do this is by looking at clusters of data instead of path-components. Intuitively, if we assume that P is sampled from some space K, clusters in the data will correspond to path-components in K. Clustering methods are built on different assumptions, but they all aim at partitioning a dataset in a similar way as we can partition a topological space into path-components.

The second adjustment we do, is to consider preimages of open sets instead of fibres. We see that since P is finite, only a finite number of fibres can be non-empty, and this would make our analogy of the Reeb graph highly disconnected.

Definition 8.3 (Mapper)**.** Let P be a point cloud, $f : P \to \mathbb{R}^m$ a filter function for some $m \in \mathbb{N}$ and $\mathcal{U} = \{U_i\}$ a finite cover of $\operatorname{Im} f$. Assume also that we have a clustering algorithm that partitions any subset $V \subseteq P$ into clusters $V_1, V_2, \ldots, V_n$.

Now, since $\mathcal{U}$ is a cover of $\operatorname{Im} f$, we have a cover of P

$$f^*(\mathcal{U}) = \{f^{-1}(U) : U \in \mathcal{U}\}.$$

By clustering each set $f^{-1}(U) \in f^*(\mathcal{U})$ into subsets we get another cover of P

$$\widehat{f^*(\mathcal{U})} = \{V : V \text{ is a cluster in } f^{-1}(U) \text{ for } f^{-1}(U) \in f^*(\mathcal{U})\}.$$

The **Mapper complex** is the nerve of this cover

$$\mathcal{N}(\widehat{f^*(\mathcal{U})}).$$

In practice, we usually talk about the **Mapper graph** which is simply the 1-skeleton of the Mapper complex.

Example 8.4. Figure 18 illustrates the Mapper pipeline. We see that the circle and the flare are both preserved in the resulting Mapper graph.

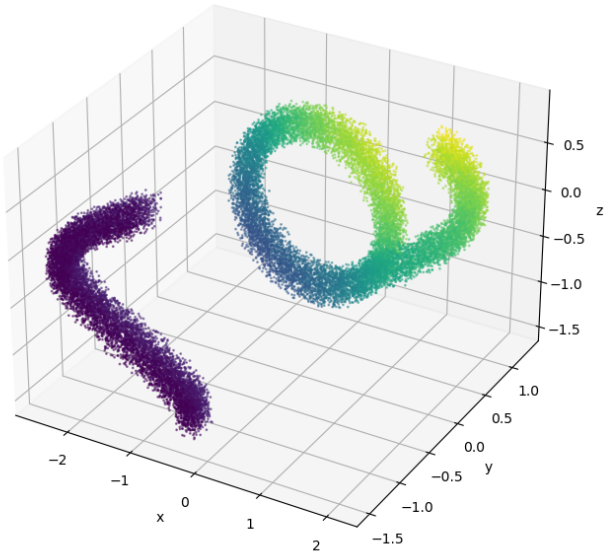


FIGURE 19. Three dimensional synthetic data, coloured by the y-coordinate value.

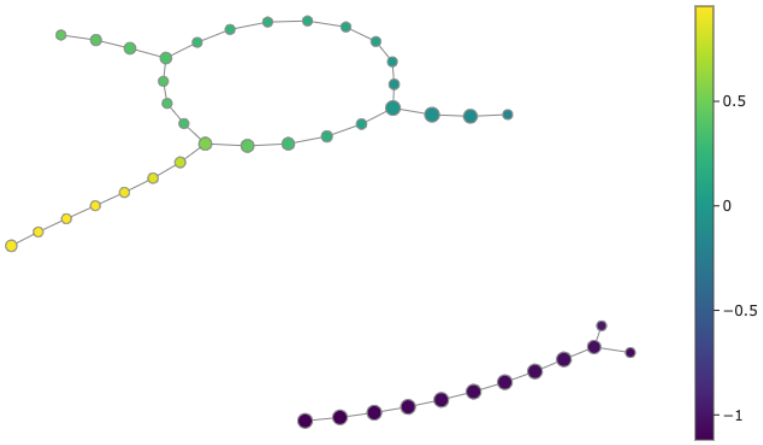


FIGURE 20. Mapper graph of the data in Figure 19. Projection on the y-axis is used as filter function. The covering is *balanced*, with 25 intervals and overlap 0.3. For clustering we use DBSCAN [16] with *eps* 0.2 and *min_samples* 9. The *min_intersection* is 1. The data is coloured by the average y-value in each node.

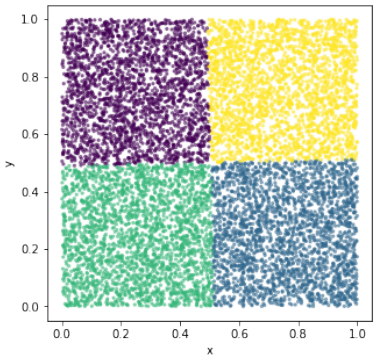


FIGURE 21. Uniform random data ($X \sim U([0,1]), Y \sim U([0,1])$) coloured with the labels found by k-means clustering with $k = 4$.

Remark 8.5. The Mapper complex is defined as a nerve. A goal for Mapper could be to recover the homotopy type of the data, as is the case if the assumptions of the Nerve Theorem (Theorem 6.3) are satisfied. These assumptions depend on what kind of clustering algorithms are used. The Reeb graph does, for example, not always recover the homotopy type of the original space, as the example in Figure 17 illustrates. It is easy to see that in this case the fibres are not contractible, and this is the reason the nerve theorem does not apply. It is not easy to do these considerations about unknown spaces, like those we typically deal with in a data setting.

When applying Mapper to a dataset we have to make many choices of methods and parameters. We need to choose a *filter function* on the dataset, a cover $\mathcal{U}$ and a way to cluster the data. In this thesis we will use projection on one or two principal components [22] as filter function. We will cover the image of the filter function by intervals or rectangles. The intervals and rectangles will either all be equal in size, we call this a "uniform" covering, or all contain the same number of points in their preimage, we call this a "balanced" covering. The intervals or rectangles must have a specified overlap, and we define this in terms of the fractional overlap with the next interval or rectangle, in terms of area of rectangles or length of intervals. We will use `scikit-learn`'s [37] k-means clustering implementation, choosing only the number of clusters. Some times we will use ToMaTo clustering [10, 1] instead of k-means. For better robustness to outliers and noise, we sometimes increase the number of points two nodes must have in common for an edge to join them. We refer to this as the *min_ intersection*. We use `giotto-tda`'s [54] implementation of Mapper.

Example 8.6. Let us apply Mapper to the synthetic data in Figure 19. The result is shown in Figure 20. We see that we get two connected components in the Mapper graph, corresponding to the "connected components" in the data. The line segment looks like a line segment in the Mapper graph as well. The loop is present in the Mapper graph, but with three flares instead of the one that we have in the original data. The short flares are due to the "thickness" of the loop. This is the same thing that happened with the Reeb graph of the torus in Example 8.2.

FIGURE 22. Mapper graph of uniform data. The filter function is projection on the two components. We use a *balanced* 8×8 cover with 0.3 overlap. For clustering we use ToMaTo clustering with $k = 50$ and $\tau = 0.5$. The *min_ intersection* is 1. The points are coloured by the average x-value of the data points belonging to the node.

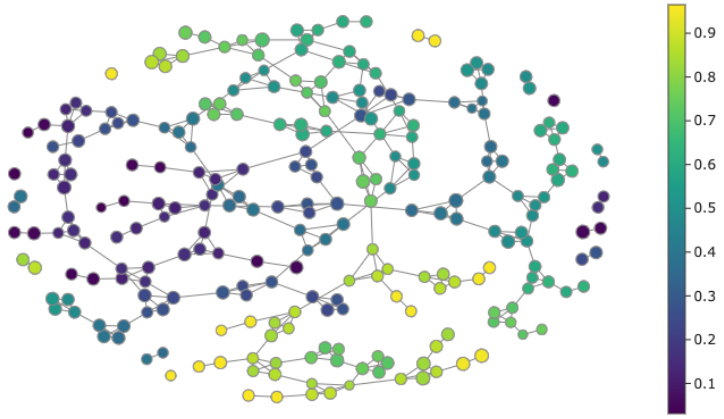


FIGURE 23. Mapper graph of uniform data. Filter function is projection on the two components. We use a *balanced* 8×8 cover with 0.3 overlap. For clustering we use k-means clustering with $k = 4$. The *min_ intersection* is 1. The points are coloured by the average x-value of the data points belonging to the node.

Example 8.7. This example illustrates Mapper's sensitivity to the clustering method. Consider the random uniform data in Figure 21. With projection on the two components as the filter function, we would expect the Mapper graph of such data to look like a grid, regardless of the other parameter choices. Consider then the colouring in Figure 21 showing the clustering obtained with k-means clustering ($k = 4$). From a topological perspective, we see that k-means clustering fails dramatically at capturing the connected components in the data, since it is only 1 such component. From the theoretical perspective, this is unfortunate for our interpretation and understanding of the Mapper graph. In practice, this behaviour introduces highly spurious shape in the Mapper graph.

In Figure 22 we see the Mapper graph obtained with ToMaTo clustering. The graph looks like a grid, like we would expect. In Figure 23 obtained with k-means clustering, on the other hand, the graph looks nothing like the data. This happens because each preimage of the rectangular covering, looks exactly like the data in Figure 21, and therefore k-means clusters the preimages of the cover in the way illustrated in the figure. This creates horizontal and vertical divides, since no points are shared between the clusters in the same preimage. From the colouring we can conclude that the Mapper graph is not completely nonsensical, since connected nodes consist of points that lie close together. The problem is just that some nodes that consist of points that are close are not connected.

It would of course be fairly pointless to apply Mapper to such simple data. However, when we deal with data which has simple shape, like that in Figure 21, we would hope that the Mapper graph reflects this. It certainly can, but it is more dependent on the clustering algorithm than we might like.

Consider a slightly more realistic setting, where the data does not lie perfectly on the plane as in Figure 21, but close. We model this as the three-dimensional random uniform data $X \sim U([0,1]), Y \sim U([0,1]), Z \sim U([0,0.2])$. The same Mapper procedure with k-means now produces the Mapper graph in Figure 24, which captures the two-dimensional manifold structure in the data. The explanation, is that even though most of the variance of the data is in the xy-plane, the variance in the z-direction is significant in the preimages of the covering. Thus, we end up with a clustering of the preimages like that in Figure 25, and while this also divides the one connected component into 4 clusters, it does not introduce the unfortunate cuts we had in the previous example.

Despite k-means poor performance at preserving the 2-dimensional data, I claim that k-means can be a useful clustering function for the Mapper pipeline. The data we work with is high-dimensional and noisy, and we therefore expect the problems we encountered in Example 8.7 to be irrelevant. Compared to other clustering methods like DBSCAN [16, 44] or ToMaTo [10] clustering, k-means clustering produces clusters of similar size. It runs faster than essentially any clustering, and it is far easier to choose the number of clusters than some sensible distance threshold like we need for DBSCAN (which gets even more difficult with high-dimensional data as we will have). With a specified number of clusters it is also easier to specify the resolution (the number of nodes) of the Mapper graph. We just need to be careful with making bombastic statements about the shape of the data based solely on Mapper graphs. This is important regardless of the clustering method chosen, since the Mapper pipeline is highly sensitive to most of its parameters and choices of methods.

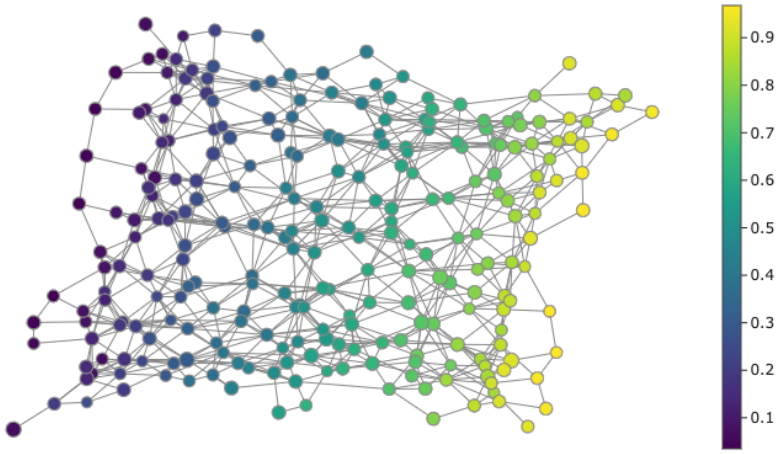


FIGURE 24. Mapper graph of three-dimensional uniform data $X \sim U([0,1]), Y \sim U([0,1]), Z \sim U([0,0.2])$. Filter function is projection on the two components. We use a *balanced* 8×8 cover with 0.3 overlap. For clustering we use k-means clustering with $k = 4$. The *min_ intersection* is 1. The points are coloured by the average x-value of the data points belonging to the node.

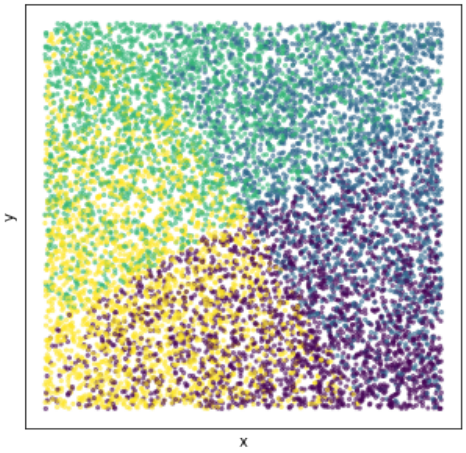


FIGURE 25. Qualitative visualisation of a preimage of the covering of three-dimensional uniform data $X \sim U([0,1]), Y \sim U([0,1]), Z \sim U([0,0.2])$, coloured by the clustering labels. The separation along the z-axis results in points with similar (x, y)-coordinates being assigned to different clusters.

9. UMAP

We want to understand the shape of the space K from which the data, i.e. the point cloud P, is sampled. If the data lies in a high-dimensional space, this can be difficult. Another approach to the problem is to try to reduce the dimensionality of the data, while preserving the topology. For example, if we can reduce it to 2 or 3 dimensions, we can visualise it and find the topology from the visualisation. Thus, we have transformed our problem into a *dimensionality reduction* problem, where we pay special attention to topological properties of the data (which is not the case for many dimensionality reduction methods).

Recall the general dimensionality reduction problem formulation: We have data P from a high-dimensional space, and we want to find a function, $f : P \to E$, where E is a low-dimensional space, typically chosen to be $\mathbb{R}^m$ for some relatively small m. We want the function f to preserve certain properties of the data, so that we can deduce things about P from the low-dimensional embedding of the data $f(P) \subseteq E$.

What we mean by "preserving certain properties" in the above formulation is one of the choices that has to be made when designing or choosing a dimensionality reduction method for an application. The variance of the data (PCA [22]), pairwise distances (MDS [24]) or local pairwise distances (t-SNE [29]) are examples of properties that many dimensionality reduction methods aim to preserve. In our case, the *topology* (or the *shape*) of the data is the property that we want to preserve. This places us in the domain of *manifold learning* dimensionality reduction. Explicitly, we assume $P \subseteq M$, that is, that the data lies on some manifold M.

McInnes, Healy and Melville introduce in [30] the manifold learning dimensionality reduction method UMAP (Uniform Manifold Approximation and Projection), which solves the problem stated above. Justifying whether a dimensionality reduction method actually preserves the manifold is in fact a far from trivial task. On the other hand, measuring the preservation of pairwise distances, either locally or globally, is a simple task, at least empirically. In order to justify the suitability of the method, UMAP is designed with strong justification from the theory of algebraic topology and fuzzy simplicial sets developed by Spivak in [51]. However, for the purposes of this thesis we will not require to understand in detail the theory that justifies the construction. We will therefore focus on how UMAP works in practice, to get an intuitive understanding. The explanation of UMAP given here is inspired by the similar description given in the UMAP documentation [25] under "How UMAP Works".

The UMAP algorithm can roughly be summarised in two steps:

(1) Create a simplicial complex from the data.
(2) Find a layout of the data in a predefined embedding space that through the (almost) same procedure produces as similar a simplicial complex as possible.

Remark 9.1. We do not actually create a simplicial complex, but a (fuzzy) *simplicial set* from the data. For the intuitive understanding we will, however, work with simplicial sets, since they are far easier to understand, in line with the description given in "How Does UMAP Work" [25].

9.1. Creating the Fuzzy Simplicial Complex.

Assume that the data P lies in $\mathbb{R}^n$ for some positive $n \in \mathbb{N}$. As previously mentioned, the Čech complex is a good

simplicial complex to associate with a point cloud, since its geometric realisation is homotopy equivalent to the union of balls P_r by the Nerve Theorem (Theorem 6.3). As before, we encounter the problem of choosing the scale r.

UMAP is built on the assumption that the data P lies on a manifold M, and that there exists a Riemannian metric such that the data is uniformly distributed on the manifold with regard to that metric. However, the Riemannian metric and the metric on $\mathbb{R}^n$ might not agree, since data is rarely uniformly distributed in $\mathbb{R}^n$. Moreover, we do not know the Riemannian metric we assumed exists. To overcome this, we approximate the Riemannian metric locally.

If we place a ball with a suitable radius on the manifold, with respect to the Riemannian metric, it should contain approximately the same number of points regardless of its location, since the data is uniformly distributed with respect to this metric. We can use this observation to approximate the Riemannian metric locally by flipping it around. For $p \in P$, a ball centred at p containing k points, should have the same volume, regardless of the point p. Hence, we can obtain a local approximation d_p of the Riemannian metric for each point $p \in P$ by scaling the metric d on $\mathbb{R}^n$ such that the distance to the k-th nearest neighbour of p is 1 in the local metric. In the Riemannian metric, a suitable scale for the Čech complex is the average distance to the k-th nearest neighbour, since then all points will be connected to approximately k other points. In the local approximation, this amounts to making a Čech complex with scale 1 in the local metrics. The 1-skeleton of the Čech complex will simply be the k-neighbour graph of the dataset; each point has an edge to all its k-th nearest neighbours.

Remark 9.2. The local metrics we have designed do not constitute a global metric, so speaking about the Čech complex at scale 1 with the local metrics is a bit imprecise. We actually mean that we create a space

$$P' = \bigcup_{p \in P} \overline{B(p; r_p)} \subseteq \mathbb{R}^n,$$

where r_p is the distance to the k-th nearest neighbour of p. The collection $\{\overline{B(p; r_p)}\}_{p \in P}$ satisfies the assumptions of the Nerve Theorem (Theorem 6.3) so the nerve of this collection gives a simplicial complex with a geometric realisation that is homotopy equivalent to P'. This nerve is what we mean when we refer to the Čech complex with the local metrics above.

Example 9.3. Figure 26 shows data lying on the circle S^1. In the left plot the data is uniformly distributed in $\mathbb{R}^2$, so the euclidean metric and the Riemannian metric agree. Here, the average distance to the third nearest neighbour provides a good scale for the Čech complex. In the middle plot, the data is not uniformly distributed in $\mathbb{R}^2$. With the same scale as in the left figure, we now fail to capture the underlying manifold. In the right plot, we see the union of balls with fixed radius, but in the local approximation to the Riemannian metric. The local approximation helps us to capture the manifold structure.

Since we have local metrics, we can weigh the edges of the graph (1-skeleton of the Čech complex), by the length of the edges. Since the local metrics are inconsistent, we can interpret the result as a directed weighted graph, where an edge from p to q is weighted with a function of the distance $d_p(p, q)$ in the local metric on p, while the edge from q to p is weighted with a function of the distance $d_q(p, q)$ in the local

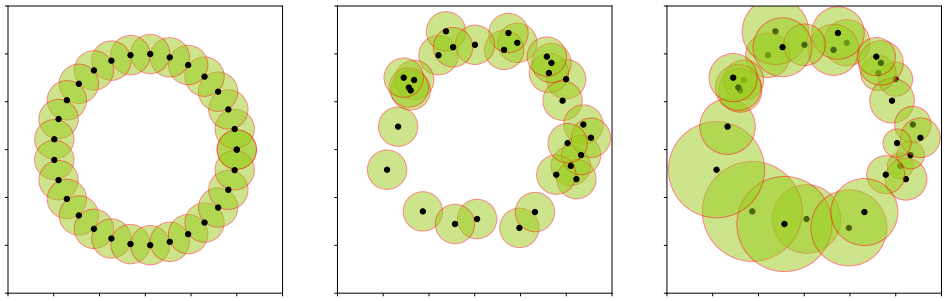


FIGURE 26. Left: Uniform data X and the union of balls X_r indicated with the green circles. Middle: Non-uniform data Y (with noise) and the union of balls Y_r indicated with the green circles. Right: Non-uniform data Y (with noise) and the union of balls Y'_r, scaled by the local density.

metric on q. Explicitly, to deal with the curse of dimensionality [57], the authors set the weight of the edge from p to q in the graph to

$$w(p,q) = \exp\left(\frac{-(d_p(p,q) - \rho_p)}{\sigma_p}\right),$$

where ρ_p is the local distance to the nearest neighbour of p, d_p is the local metric on p and σ_p is a constant normalising the sum of the weights of all the edges originating at p.

So far, we have created a weighted directed graph. To obtain an undirected graph, we replace each pair of edges (p,q), (q,p) and weights $w(p,q)$, $w(q,p)$, with an undirected edge (p,q) with weight given by the fuzzy union

$$w'(p,q) = w(p,q) + w(q,p) - w(p,q)w(q,p).$$

If we interpret $w(p,q)$ and $w(q,p)$ as the probabilities that the directed edges (p,q) and (q,p) exist, respectively, the new weight $w'(p,q)$ is the probability that at least one of the directed edges exist. We have thus obtained a fuzzy simplicial complex (or a weighted undirected graph), representing the dataset, like in Figure 27.

Remark 9.4. It is possible that we have an edge (p,q), but not an edge (q,p) in the directed graph, and in this case the above definition does not make sense. We resolve this by setting the weight $w(q,p) = 0$ if the edge (q,p) is not present in the graph, very much in line with the probability interpretation of the weights.

The theory presented in [30, Section 2], justifies the choices taken above and proves that the fuzzy simplicial complex (set) we have created faithfully captures the topology of the space we are investigating. I refer the reader to the discussions in the paper for details.

Remark 9.5. There are no limitations to generalise the above procedure to consider higher-order simplices in the fuzzy simplicial complex. However, the authors of UMAP have decided that they see little gain in accuracy and a high computational cost of doing so. Therefore, the current implementation of UMAP [25] considers only 1-simplices.

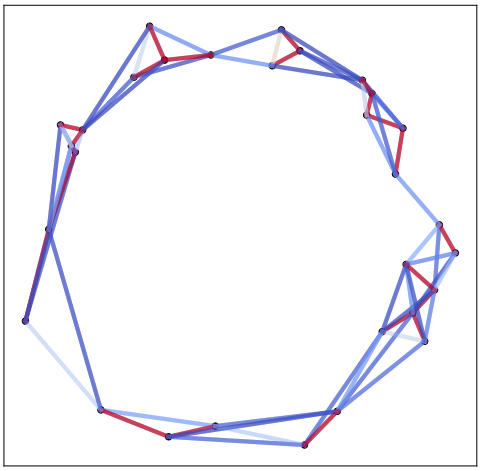


FIGURE 27. Fuzzy simplicial complex (1-skelelon) obtained from the point cloud Y from Figure 26. The edges are coloured according to their weight (red is high, blue is low).

9.2. **Optimising a Layout.** Now we want to find a layout of the dataset in an embedding space E of our choice that has a similar representation as the one we computed for P in the previous subsection. The embedding space E could for example be $\mathbb{R}^2$ or $\mathbb{R}^3$, but in some special cases it might be suitable to embed the data on the sphere S^2, the torus $S^1 \times S^1$ or some other space. If we apply dimensionality reduction as a preprocessing step before applying machine learning techniques to the data, we might sometimes want to embed the data in $\mathbb{R}^n$ for fairly large n.

Given an embedding of the data P in the embedding space E, that is, a function $f : P \to E$, we can find a fuzzy simplicial complex of the point cloud $f(P)$ with the same procedure as in the previous subsection. However, we want the embedding to have a fairly uniform distribution and a globally sensible metric. Therefore, we leave out the step of finding the local metrics based on the local density estimates. The distance to p's nearest neighbour ρ_p is also a size we would want to be fairly similar for all the points in the embedding. Hence, we give this value as a hyperparameter. This gives us a procedure to obtain a fuzzy simplicial complex from the embedding.

We now need a measure of similarity between the two fuzzy simplicial complexes. The creators of UMAP argue that in line with the probability interpretation used earlier, the suitable measure of similarity is the cross-entropy [56]. Let $w : P \times P \to \mathbb{R}$ be the weight function of the fuzzy simplicial complex of the original data P and $u : P \times P \to \mathbb{R}$ the weight function of the fuzzy simplicial complex of the embedding $f(P)$. Then, the loss function becomes

$$\sum_{p,q \in P} \left[w(p,q) \log\left(\frac{w(p,q)}{u(p,q)}\right) + (1 - w(p,q)) \log\left(\frac{1 - w(p,q)}{1 - u(p,q)}\right) \right].$$

This function can be made differentiable with respect to the layout in the embedding space. The layout can then be optimised with stochastic gradient descent. This yields the UMAP algorithm.

10. The Space of Activations

We are going to explore the topology of the space of activation vectors in an intermediate layer of a convolutional neural network (CNN). Specifically, we will explore these activations obtained by feeding patches from the ImageNet dataset to the CNN known as GoogLeNet. For completeness, we give a brief description of neural networks and convolutional neural networks here.

10.1. Neural Networks.

Definition 10.1. A **neural network** Φ is a finite sequence of functions $\mathbb{R}^n \to \mathbb{R}^m$ of the form

$$v \mapsto \varphi(Av + b),$$

where $A \in \mathbb{R}^{m \times n}$ is the **weight matrix**, $b \in \mathbb{R}^m$ is the **bias** and $\varphi : \mathbb{R} \to \mathbb{R}$ is a continuous non-linear function. We call φ the **activation function** and interpret $\varphi(Av + b)$ as applying the function φ element-wise to the vector $Av + b$.

Each function is called a **layer**. The weight matrix, bias, activation function and dimensions n, m can of course be different for the different layers in the network.

The neural network defines a function $\Phi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ from an input space to an output space, where n_x and n_y are the dimensions of the input space and output space, respectively.

Remark 10.2. Without the non-linearity of the activation function, the network Φ would simply be a linear function. In practice ReLU (Rectified Linear Unit),

$$x \mapsto \max(x, 0)$$

is the most common activation function. The activation function in the final layer is commonly chosen to be different from the activation functions in the intermediate layers, depending on the purpose of the neural network. With ReLU as activation function, sufficiently large neural networks are capable of approximating essentially any function $\mathbb{R}^n \to \mathbb{R}^m$ for arbitrary $n, m \in \mathbb{N}$, cf. survey of expressivity of neural networks [18].

Neural networks can be used to model complex relationships in data. Given samples of data and a target variable, we can iteratively optimise the weight matrices and biases of the neural network to obtain a function that predicts the target, if given data. The optimisation is done with variations of gradient descent. See [3] for more about neural networks and *training* of neural networks, that is the optimisation of the weights in order to make the network model the relationship between input and output data. Neural networks' capability of expressing essentially all kinds of relationships [18] makes them useful in a wide variety of tasks. Moreover, overparametrised neural networks, that is neural networks with many layers and/or high dimensionality of the layers, achieve state-of-the-art performance on a huge variety of machine learning tasks, such as ImageNet [13], the NLP system GPT-3 [5] and the protein structure predictor AlphaFold [23].

10.2. Convolutional Neural Networks.

A convolutional neural network (CNN) is a neural network with one or more convolutional layers, designed to pay attention to the format of the input data. The most common application is on image data, and we will explain convolutional neural networks designed for image data only. The CNN that we will investigate was designed to do image classification. For example, if we feed an image of a dog to the neural network, we want the network

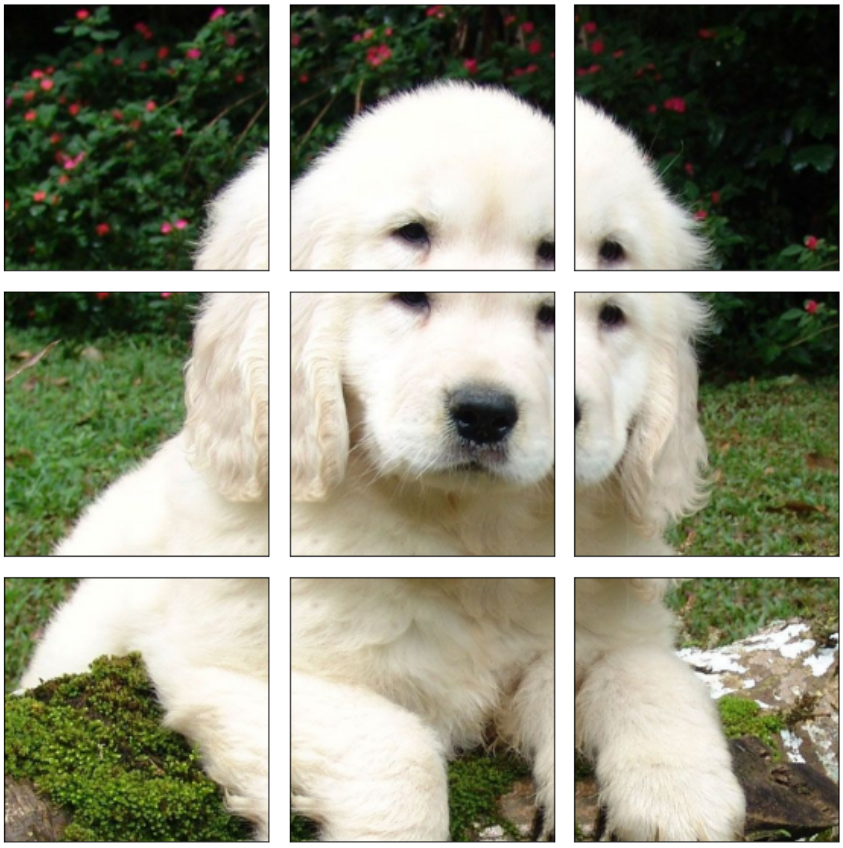


FIGURE 28. An image from the ImageNet database [13] split into overlapping patches. In a CNN, the size of patches, specified by the dimension of the kernels, are typically much smaller than in this example.

to output the label corresponding to "dog", or maybe the particular breed of dog like "English Setter". In order to train the network to classify the images into k distinct classes we have data in the form of pairs (x, y), where $x \in \mathbb{R}^{n \times m}$ is an image (in this case grayscale) and $y \in 1, 2, 3, \ldots, k$ is the class the image belongs to, e.g. "dog".

We could make a neural network as in Definition 10.1 by considering the image $x \in \mathbb{R}^{n \times m}$ as a vector in $\mathbb{R}^{nm}$, simply by flattening the image, and train a neural network to classify the images, by considering the labels $\{1, 2, \ldots, 1000\}$ as a subset of $\mathbb{R}^{1000}$, for example by doing one-hot-encoding [61]. The problem with this approach is that we lose information about the relative positions of the pixels in the image. Another practical problem is that the space $\mathbb{R}^{nm}$ is very high-dimensional and the task of classifying images is quite complex, so we would require very large weight matrices and many layers.

A way we could deal with the high dimensionality would be to split the image up into patches and train a network that takes smaller images as inputs. This could

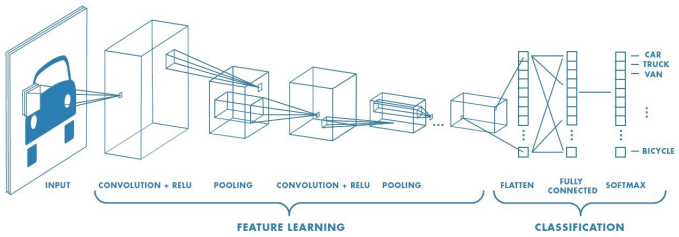


FIGURE 29. An image classification CNN architecture. Image from [52].

greatly reduce the number of parameters required to train in the neural network. One could also argue that this approach makes sense, since it enforces the "nearness" between pixels in the same patch, in the sense that what the network sees in the upper right corner should not depend on a pixel value in the lower left corner. In addition, a classification of an image should not depend on the location of the dog in the image, and therefore it makes sense to apply the same network to all the patches. In this way, we could make the classifier more robust to translations of the images. Since we do not know exactly the location of the patterns we are looking for in the image, we might also want to make overlapping patches from the image, like in Figure 28.

The same argument could be applied to the task that the first layer in a neural network tries to perform, even if we do not know exactly what task this is. Whatever the task is, the network should not care about a pixel in the lower left corner when looking at the upper right corner. Therefore, each *layer* should look at patches independently of each other. This is exactly what a convolutional layer does. Moreover, the output of a convolutional layer can be understood as a stack of images or an image with many channels, and this allows us to let convolutional layers follow convolutional layers in a neural network.

The CNN exploits the relative positions of the pixels. This is actually just a sparse version of the general neural network from Definition 10.1, where we force a large number of entries in the weight matrices to be zero as well as enforcing groups of weights to have the same value. It is, however, more convenient to describe the convolutional neural network using different notation.

Definition 10.3. (Convolutional Layer) A **convolutional layer** is a function $f : \mathbb{R}^{n\times m\times l} \to \mathbb{R}^{n'\times m'\times l'}$. The function is determined by l' **kernels**, the **bias** $b \in \mathbb{R}^{n'\times m'\times l'}$ and the **activation function** $\varphi : \mathbb{R} \to \mathbb{R}$. A kernel of size s is an element in $\mathbb{R}^{s\times s\times l}$. For $x \in \mathbb{R}^{n\times m\times l}$, the output at the index (i, j, p) is

$$f(x)_{i,j,p} = \varphi(\langle k, x'\rangle + b_{i,j,p})$$

where k is the p-th kernel in the convolutional layer, $b_{i,j,p}$ is the bias at index (i, j, p). The inner product is the usual dot product and $x' \in \mathbb{R}^{s\times s\times l}$ is defined by

$$x'_{i',j',p'} = x_{i+i',j+j',p'},$$

that is a part of the output of the previous layer indexed by i' and j'.

Remark 10.4. Other parameters are often used to define a convolutional layer, such as stride or padding, and a pooling layer is often added after convolutional

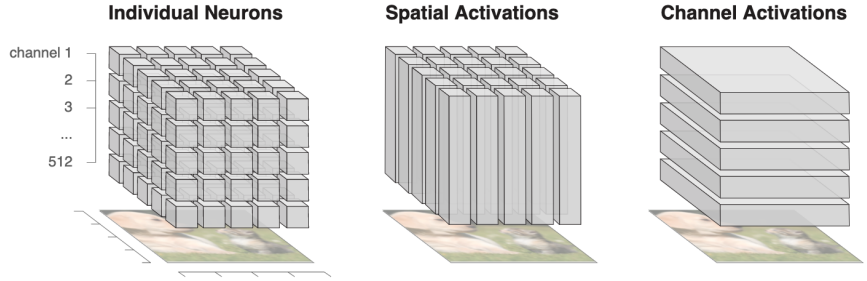


FIGURE 30. A cuboid of activations. Illustration taken from [35].

layers. We will not concern ourselves with details of what happens at the boundaries of the images or considerations relating to stride and pooling. These techniques are present in the network we are investigating, but the details of these techniques are assumed to be of little relevance to the spaces we are studying. See the book by Aggarwal [3, Chapter 8] for more details.

Remark 10.5. A convolutional layer can be understood as a transformation of data in the shape of a cuboid, as illustrated in Figure 30. A stack of l images represents the input of a convolutional layer. An RGB image can for example be represented as a stack of three images, one of each colour.

In line with the inspiration taken from the brain, we refer to each "entry" or "pixel" in the cuboid as a *neuron*. If we have a high value at a neuron at a particular layer, we say the neuron has a high *activation*.

We will refer to the position in the images as the *spatial activations*. For example, let us say we have a 28×28 RGB image, represented as a stack of three images. The spatial activation, say $(4, 8)$, is then represented as a vector of three elements, or three neurons, corresponding to the three colours.

We refer to each "image" as the *channel activation*. In the RGB example we have three channels, as illustrated in Figure 31. Two neurons in the same channel, but at different spatial locations are activated by the same patterns in the input image, but placed at different locations. In this way, the activations in a channel tell us "what" the network sees, while the spatial activations tell us "where" in the image the network sees things. This information is what allows the network to combine different patterns at different locations into even more complex patterns in the next layer.

A convolutional neural network usually consists of a number of convolutional layers, followed by some fully connected layers, like the layers defined in Definition 10.1. Figure 29 shows an example of a typical convolutional neural network architecture, visualised as the format of an image as it passes through the layers of the network. For the convolutional layers, the input and output data format is a cuboid of activations or a stack of images, as in the "feature learning" part of the figure. In the "classification" part of the network, the data is in the format of vectors, as in Definition 10.1.

In the image classification setting, the convolutional layers do in a sense detect general features in the images, while the later fully connected layers classify the

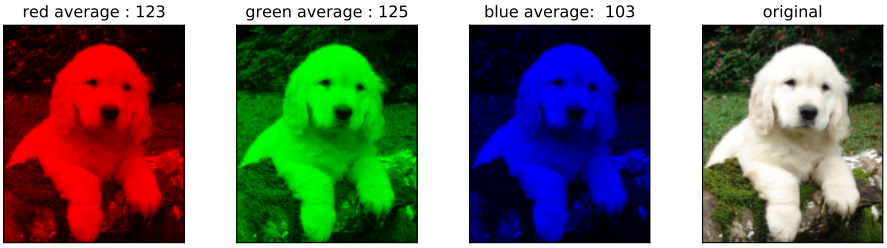


FIGURE 31. The channel activations of an RGB image. The average activations show that green is the most prominent colour in the image, while blue is the least prominent colour. Original image on right hand side is taken from the ImageNet database [13].

images based on the features. The difficulty of understanding convolutional neural networks stems in part from the fact that during training the network simultaneously learns both the feature representation, as well as how to classify the images based on the features. In a simpler setting, we could first learn features of the images and then train a classifier when the features are already learned. A simple example could be to use a number of the principal components [22] of the data as the features and then train a logistic classifier [59] on these features. In this case, it is easier to understand the features we create. In a convolutional neural network, the features are in a sense specialised for the task the network is trained to do, as well as to the data it is trained on. This usually makes the features very useful, but also difficult to understand.

Let us say that we have trained a convolutional neural network to classify images of cats and dogs. It is not a trivial task to create good features for such an application. It is also not easy to even understand what separates a good feature from a bad feature. Let us say that we noticed that one of the features of the network seems to identify whether or not there is a bowl of milk in the image, and this increases the networks certainty that the image is of a cat. Whether this is a good or a bad feature depends on the application we have in mind. For example, if the network is intended to classify dogs and cats for a self-driving car, the feature is probably completely useless. If it is intended to automatically tag images on social media, the bowl of milk is probably a very useful feature. The knowledge that the network we have trained sees bowls of milk as a feature is, however, very useful information in both cases. The issue is that this kind of understanding of the network is very hard to obtain. Past and ongoing research aims to create methods for getting a better understanding of the features created by convolutional neural networks. See for example [35, 34, 9].

10.3. **GoogLeNet.** We will investigate the convolutional neural network called GoogLeNet that was introduced in the paper [53] and won the 2014 ImageNet Large Scale Visual Recognition Challenge [42]. The architecture is shown in Figure 32. The network is built around the Inception module, which is just a concatenation of convolutional layers of different strides in a specific design. See [3, Chapter 8.4.4] for details. In GoogLeNet, the inception modules are named "3a", "3b", "4a" and so on. For our purposes it suffices to view the inception modules as a slightly more

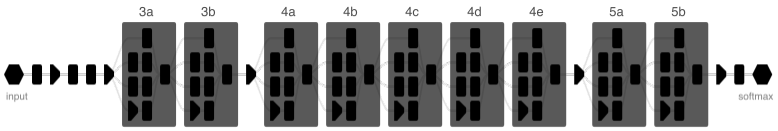


FIGURE 32. The architecture of GoogLeNet. Illustration from [9].

complicated convolutional layer. The triangles in the figure are pooling layers, reducing the spatial resolution of the data propagating through the network.

10.4. **Space of Activations.** Since a convolutional neural network is simply a sequence of functions $\mathbb{R}^{n\times m\times l} \to \mathbb{R}^{n'\times m'\times l'}$, we can investigate the *activations* at an intermediate layer of the network. For example, we can look at the activations after transforming the image up to the "4b" module, but before passing it to the "4c" module. Let $I = \mathbb{R}^{256\times 256\times 3}$ be the space of 256×256 RGB images. There are multiple choices of metrics on this space, for example we can flatten the images and use euclidean distance between the vectors. Formally, the sequence of functions defines a function $\Psi_{4b} : I \to \mathbb{R}^{n\times m\times l}$, taking images as input and returning the activations after the "4b" module. We can of course define $\Psi_{3a}, \Psi_{3b}, \ldots, \Psi_{5b}$ similarly. For an image $x \in I$ the activation $\Psi_{4b}(x)$ is an element of $\mathbb{R}^{n\times m\times l}$. For a set of images P we obtain a *point cloud of activations* $P_{4b} = \Psi_{4b}(P) \subseteq \mathbb{R}^{n\times m\times l}$. If we assume that there exists a space $K \subseteq I$ of "real" images, we can define $K_{4b} = \Psi_{4b}(K)$. By assumption, we have $P \subseteq K$ and thus $P_{4b} \subseteq K_{4b}$. The aim of the remainder of the thesis is to investigate the shape of the space K_{4b} from the samples P_{4b}.

Remark 10.6. Instead of assuming a space K of "real" images, we could assume a space of "likely" images. For example, if we can assign a probability density function $p : I \to \mathbb{R}$ describing the probability of encountering an image from different regions of I, we could define $K = \{x \in I : p(x) > c\}$, for some cutoff parameter $c > 0$. In this case we do not have $P \subseteq K$ and $P_{4b} \subseteq K_{4b}$, but we have that the probability that an image $x \in P$ lies in K, and consequently its activations $\Psi_{4b}(x)$ in K_{4b}, is high (depending on c).

The point cloud of activations P_{4b} is a subset of the space $\mathbb{R}^{14\times 14\times 512}$, which has dimension 100352, and for the other layers we have a similarly high dimensionality. In order to work in a more approachable dimension, we argue that we are not particularly interested in the spatial location of the activations, but rather in the kind of patterns that the CNN looks for and finds. The spatial location is of course important in order for the network to combine features in the subsequent layers, but for our analysis of the space of activations, we will disregard it completely. We argue that this allows us to focus on how the network sees different concepts in images.

As discussed, each image corresponds to a cuboid of activations $p \in \mathbb{R}^{n\times m\times l}$ in an intermediate layer, e.g. $P_{4b} \subseteq \mathbb{R}^{14\times 14\times 512}$. Explicitly, we obtain a more suitable space by sampling an activation vector at a random spatial location in p, that is, we obtain the vector at the spatial position indexed by i and j

$$(p_{i,j,1}, p_{i,j,2}, \ldots, p_{i,j,l})^T,$$

where the entries now are the channel activations. See Figure 30 for a visualisation of the cuboid of activations. Sampling one such vector from each cuboid of activations in P_{4b}, we obtain a point cloud $P'_{4b} \in \mathbb{R}^{512}$, with a dimensionality that is far more approachable. We do the same for the other point clouds $P_{4a}, \dots, P_{5b}$. We could claim that we now investigate what combinations of channel activations appear in real images. For the rest of the thesis we refer to the the point clouds $P'_{4a}, P'_{4b}, \dots, P'_{5b}$ when we write $P_{4a}, P_{4b}, \dots, P_{5b}$. For example, by P_{4b} we mean the point cloud where we have reduced the dimensionality, so $P_{4b} \subseteq \mathbb{R}^{512}$.

Since we are not particularly concerned with the length of the vectors we are dealing with, but rather with their directions, we will later normalise their length. This will be discussed shortly. For this reason, we will usually refer to vectors of activations with elements corresponding to channel activations as *directions*.

Remark 10.7. We could have taken a more sophisticated approach to obtaining a space that allows us to investigate how the network sees different concepts.

I suggest two steps in order to achieve this. Firstly, average out the spatial location of the points in P_{4b} and obtain points in $\mathbb{R}^{512}$. That is, for each channel in Figure 30 we replace the channel with the average value in the channel. Secondly, instead of passing images from the space of RGB images I, pass small patches. Explicitly, for each image $x \in I$, we randomly choose a patch of some size, e.g. 80×80, and zero out the pixels in the rest of the image, potentially adding noise to the zeroed out pixels.

Why should we look at patches instead of images? Or in other words, why do we bother with step 2? Imagine we passed images and then averaged out the spatial locations. Say, somewhat simplistically, that channel 4 detects cats and channel 120 detects dogs in the $4b$-layer. If we then passed an image $x \in I$ showing both a cat and a dog, the activation $\Psi_{4b}(x)$ would have a high activation at the 4-th channel at one spatial location and a high activation at the 120-th channel at another spatial location. After averaging we would have high activations in both the 4-th and the 120-th channel. Such a direction is unlikely to ever occur in the same spatial location, but when we look at whole images, essentially all combinations of directions could be plausible. By superimposing different patches, which is exactly what constitutes an image, we would get activations in the whole space $\mathbb{R}^{512}$, and the little information we could retrieve would only tell us what kinds of objects and patterns that usually occur together in images, but at different locations. We could for instance probably conclude that dogs occur frequently together with grass and infrequently together with water. This is not the kind of information we are looking for, so only performing step 1 would produce an unsuitable space to investigate.

What happens when we combine step 1 and step 2? Zeroing out most parts of the images ensures that distinct objects or patterns, such as dogs and cats, do not appear in the same image and thus the same activation. In effect, we are looking at what kind of directions, that is, combinations of channel activations, that appear in the same (or close) spatial locations. The hypothesis is that some directions do not exist, or are extremely uncommon.

We achieve much of the same by sampling the cuboid of activations at a random spatial location, as we do in this thesis. These points surely tell us about what kinds of directions that are present in $\mathbb{R}^{512}$. One disadvantage of this approach is that the exact interpretation of spatial activations is somewhat tricky. For example, a dog face might produce high activations corresponding to eyes and fur at *similar*

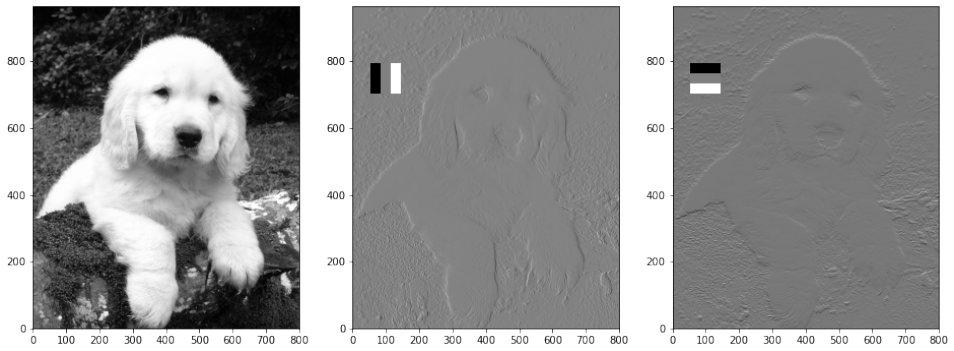


FIGURE 33. Left: sample image from imageNet. Middle: activations after convolution with the vertical edge patch. Right: activations after convolution with the horizontal edge patch. Patches are visualisations of the convolution kernels.

spatial locations, but not *exactly* at the same spatial locations. In addition, the field of reception, that is the part of the input image that a certain neuron can react to, is very large for layers deep into the network, and this makes the interpretation difficult. The two steps described above would give similar results to the random spatial sampling, but with some added translational flexibility and well-defined fields of reception.

We do not implement this way of obtaining the point cloud in this thesis, due to the significant workload and computational resources required to collect such activations. The `lucid` [45] library provides easy access to the random spatial location activation point cloud, and therefore we simply use this. Whether or not the suggested approach would yield qualitatively different results remains an open question.

10.5. **Feature Visualisation.** As previously mentioned, substantial research has been dedicated to investigate the workings of CNNs. One of the major achievements has been in the field of feature visualisation. In its simplest form, feature visualisation is about making suitable visualisations of *how the network sees an image* or *what the network looks for*.

Let us start with a simple example to get the intuition for feature visualisation. For simplicity, assume that we have trained a CNN to classify grayscale images. The first layer of the network is then quite easy to visualise. We see that since a grayscale image lies in $\mathbb{R}^{n\times m\times 1}$, the kernels of the first layer lie in $\mathbb{R}^{s\times s}$, where s is the stride. These can be visualised as patches like the ones shown in Figure 33. The convolution operation of the first layer is thus easily understood as the creation of one image as in Figure 33 per kernel. The resulting images show how well each part of the original image matches the respective kernel. In the example in Figure 33 the kernels detect vertical and horizontal edges in the image. It is well understood that the first layer of well-trained CNNs is mainly concerned with detecting edges, and potentially colours. This aligns well with the present understanding of the visual cortex of mammals, which is part of the inspiration for the CNN design in the first place.

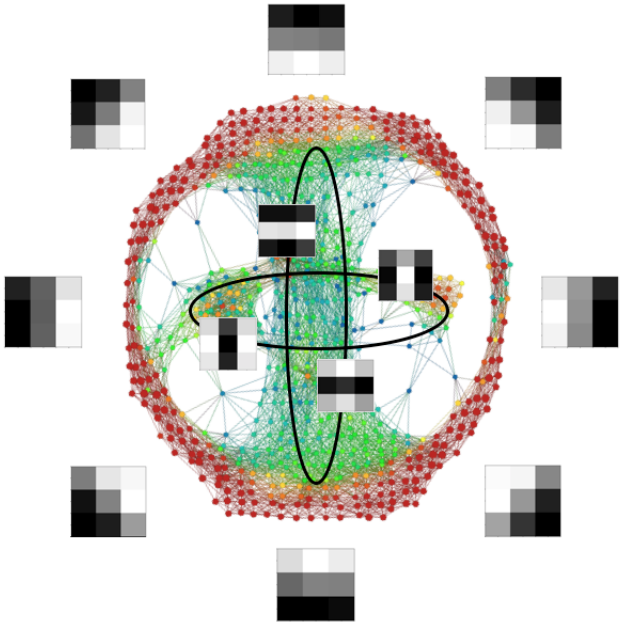


FIGURE 34. Mapper graph from spatial patches in the first layer of a CNN trained on the CIFAR-10 FINN REF dataset. FROM CARLSSON [7].

Example 10.8. Carlsson et al., [8], [6], identified edges of different directions as the most common patches that occur in natural images, and that these patches form a circle, parametrised by the angle of the edges. Moreover, they found two secondary circles that seem to detect edges that are not gradients across the patches, but in some ways sharper. They identified the shape of the overall space as a Klein bottle and parametrised the three-circle model that represents the most commonly occurring patches. More recently, Carlsson and Gabrielsson [7] found the same primary circle as well as the secondary circles in the kernels of the first layer in well-trained convolutional neural networks. See Figure 34.

The kernel that gets a maximal activation from a patch is the kernel that is equal to the patch. Thus, that the weights exhibit the same structure as the patches in natural images fits our intuition. The kernels are simply looking for what is actually present in the images. Intuitively, kernels that look for patterns in the images that simply do not occur are useless. This suggests a general approach to improving CNNs: One could study the shape of the input space and design the architecture and the weights accordingly.

This is done by Love et al. [27]. The kernels do not fill the entire space of $\mathbb{R}^{s\times s\times 1}$, but a subspace that can be parametrised. The authors create CNNs where the first layer kernels are parametrised to lie on the primary circle or on the three-circle model parametrised by the Klein bottle (with some added flexibility). This results in significant speedup in training, as well as significant increase in generalisation across datasets.

This example illustrates how feature visualisation can help us make sense of the structures that we find in data. The fact that we can visualise the patches on the primary circle found in natural images strengthens the result, since the we can see that the circle corresponds to a rotation. In addition, the visualisation allows us to conclude that the primary circle in the patches of natural images is the same as the primary circle found in the kernels of the convolutional neural networks. These considerations become significantly harder in subsequent layers.

The discussion involving Figure 33 would become slightly more complicated if we had RGB images, since then the kernels would lie in $\mathbb{R}^{s\times s\times 3}$ instead, and would therefore not be as easy to visualise. A kernel could for example detect vertical edges in the red image, but horizontal edges in the blue image, and it is not trivial to understand what convolution with such a kernel would produce. In deeper layers in the convolutional network, the kernels lie in $\mathbb{R}^{s\times s\times l}$, where l typically is larger than 100. Visualising this kernel in a sensible way is certainly challenging. It also makes matters even more complicated that for the deeper layers of the network, we do not even have a good understanding of the stack of images the kernel is convolved with, like we did in the first layer.

A successful approach to these challenges has been to visualise what the parts of the network are looking for in the image space. Explicitly, let us say we want to better understand the neuron at the spatial location $(5, 5)$ of the 9-th channel in the 4-th layer of a convolutional network. We can try to visualise what this neuron looks for in the input space, since this is interpretable as an image. We want to generate a synthetic input image that maximises the activation of this particular neuron. This is naturally phrased as an optimisation problem. Using techniques like regularisation, frequency penalisation, parametrisation and transformation robustness on the optimisation problem we get interpretable visualisations of what kind of images that activate the particular neuron.

In the same way that we can find images that activates a single neuron, we can also find images that activate a set of neurons, for example all the neurons in a channel. In this way we can visualise what kind of patterns that activates a channel, without paying attention to the spatial location of such a pattern.

The neurons (and channels) represent the directions of the standard basis for the space of activations $\mathbb{R}^{m\times n\times l}$. It is not unreasonable to believe that these directions carry some special meaning, but other directions in this space might also be meaningful. In particular, the directions that correspond to images in the dataset are of particular interest. These directions can also be visualised in the same way as the neurons, channels or general sets of neurons. Explicitly, when passing an image to the network, the activations in a layer will be a vector in $\mathbb{R}^{m\times n\times l}$. We can visualise this direction as a synthetic image that maximises the activation projected onto that direction. In a similar way we can look at the direction in the space $\mathbb{R}^l$, obtained by averaging out the spatial dependency in the channels, and visualise this direction as an image. This last procedure will be used extensively in this thesis. See [34] for more details on feature visualisation. The visualisations presented in this thesis are all created with the `lucid` python library [45].

10.6. **Normalisation.** The vectors in the point cloud P_{4b} have different lengths. The histogram in Figure 35 resembles a chi distribution, in line with an assumption that the value of a single component of the activation vector follows a normal distribution.

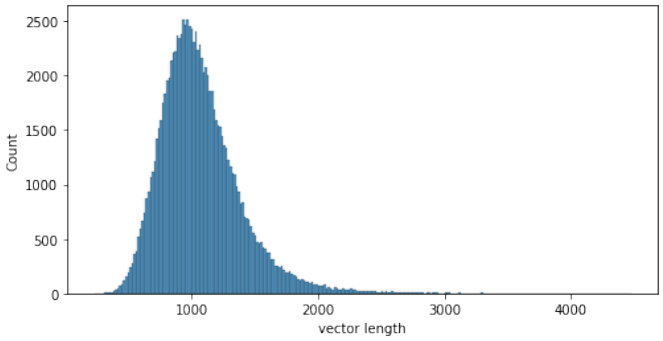


FIGURE 35. Histogram of the length of the activation vectors from the point cloud P_{4c}.

In the paper on Activation Atlases [9], the authors mainly use cosine similarity as their similarity measure, implicitly normalising the lengths of the activation vectors. For the remainder of this thesis we normalise the activation vectors, so we consider only their directions, not their lengths. The main reason for doing this, is that the feature visualisation methods we introduce in the next subsection produce visualisations of the directions, disregarding the length of the vectors.

A relevant question is whether two images of different concepts could result in activation vectors with the same direction but with different lengths. The research on feature visualisation seems to build on an implicit assumption that this is not the case, since the visualisations usually aim to maximise the activation in a direction, not to match the length of the actual vector. Essentially, we end up with the assumption that the direction specifies the concept, while the length of the vector says something about certainty or match with the concept. For example, one might expect an image and a blurred version of the same image to produce activation vectors with the same direction, but with different lengths. This argument does also to some extent justify normalisation of the vectors.

Given the assumption above, do the lengths of the vectors carry any information about the network, or do they only express a "certainty" of the detection of a concept?

We visualise the difference between vectors in P_{4c} with different lengths in Figure 36 via the following process. We bin the vectors in P_{4c} in 10 bins of ascending order according to their unnormalised lengths. We cluster each bin with k-means clustering, and from each cluster we take a random representative that we visualise. The rows in Figure 36 correspond to the bins of the data, while the columns correspond to the clusters. Hence, the upper row consists of visualisations of short vectors, while the bottom row consists of visualisations of long vectors. Recall that the visualisations are visualisations of directions and hence do not differentiate between different lengths of the vectors we visualise.

Remark 10.9. We apply k-means clustering to the bins, in order to choose representatives from the data that captures some of the variability within each bin.

There seem to be some discernible patterns in the figure. Consider, for instance, the visualisations of greenery. They seem to be present only in the upper rows, meaning that vectors in such directions typically are short. On the other hand, fur,

FIGURE 36. Visualisations of representatives from clusters (k-means) in the activations P_{4c}, binned in 10 bins according to the length of the unnormalised vectors. Each row corresponds to one bin of vectors with similar length. The rows are sorted in ascending length of the vectors. The columns correspond to the clusters within each bin.

eyes, snouts and human-looking features seem to predominantly lie in the bottom rows, indicating that vectors in these directions typically are long. These patterns indicate that some directions typically have longer activation vectors than others.

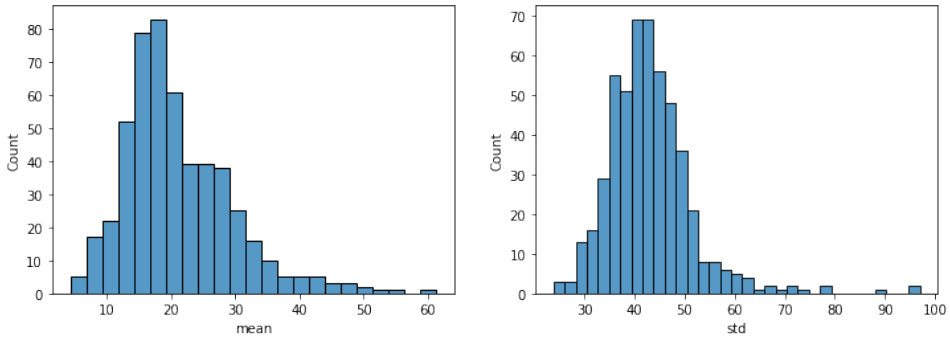


FIGURE 37. Histograms of values of mean and standard deviation for each channel in the data P_{4c}, unnormalised. That is for each channel (of the 512) we compute the mean and standard deviation in the data P_{4c}. The plots are histograms of the 512 mean values (left) and 512 standard deviation values (right).

The same phenomenon is illustrated in Figure 37. Some channels, which are just some specific directions, tend to have higher means and standard deviations. One could hypothesise that the network in some sense represents the relative importance of concepts in the size of the activations. This brings us to the next subsection.

10.7. **Direction Attribution.** Another branch of research in the explainability of CNNs is related to *attribution*, that is, to what extent do the neurons, channels, spatial locations or directions *matter*. For the CNN, which is trained to do classification, something *matters* if it is important for the classification. Quantifying this is again non-trivial and an ongoing field of research is dedicated to this task. See for example the papers on saliency maps [48] and Grad-CAM [46]. We will intentionally leave how to measure *importance* vague, since we are mainly interested in the high-level discussions involving attribution. We will not perform any attribution computations in this thesis, but the concept is introduced to get a more complete picture of the explainability research on CNNs.

Specifically, what we want is a function $a : P_{4b} \to \mathbb{R}$ that measures the importance of each direction in P_{4b}, or the point cloud in other layers. Since we are not looking at the complete images, but rather at random spatial locations, we want to measure how much the network cares about what it sees in a specific location. For example, a patch showing grass is probably not given much significance by the network, while a patch with a pair of eyes probably is. If we can measure the significance, we could focus our attention on the subspaces that are of high significance.

One could also have attribution functions that measure the importance of a direction for a classification of a specific label, e.g. how much does this direction contribute to a hypothetical classification of an image as "grey whale". We refer to the latter as *class attribution*.

Meaningful functions on the activation vectors give good filter functions for the Mapper algorithm, as well as allowing for additional manipulation and better interpretation of the point cloud of activations.

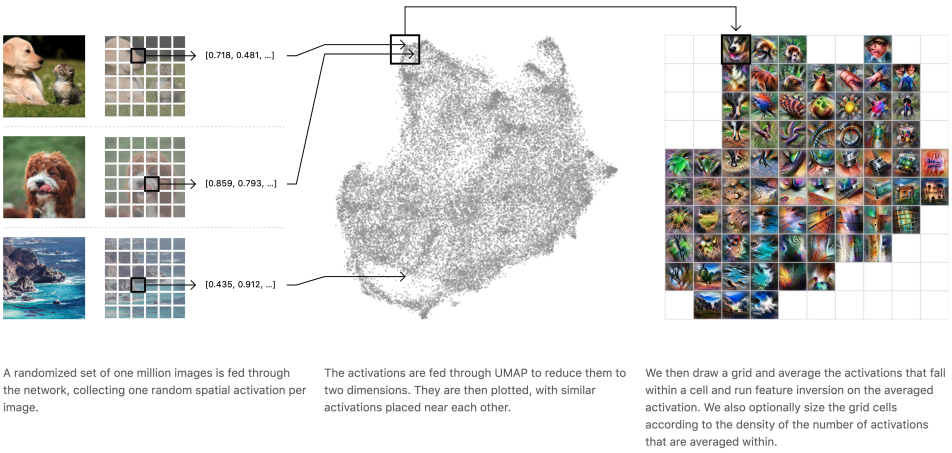


FIGURE 38. Creation of an Activation Atlas. A random spatial activation is sampled for each image presented to the network. UMAP is applied to the space P_{4c} to obtain the middle scatterplot. The points are grouped into grid points. For each grid point we create a visualisation of the average activation within that point. Figure taken from [9].

11. Activation Atlases

In [9] the authors create an *Activation Atlas* from the point cloud of activations, e.g. P_{4c} from layer 4*c*. The Activation Atlas aims to summarise information about the point cloud P_{4c} in an accessible format. A brief explanation will follow. For a more thorough introduction to the Activation Atlas I refer the reader to the original paper [9].

The creation of an Activation Atlas is described in Figure 38. UMAP is applied to the point cloud, e.g. P_{4c}, to get a projection to $\mathbb{R}^2$. The subspace in $\mathbb{R}^2$ populated by the UMAP projection is then divided into a grid. We take the average activation within each rectangle in the grid and visualise the average activation. The visualisations are then arranged in the grid, so that concepts considered similar by the network hopefully are placed close together.

Remark 11.1. Note that in the paper [9] they use one million activations to create their atlases. In addition, the parameters of the UMAP dimensionality reduction are not necessarily the same as in the paper, since they are not explicitly stated. This is assumed to be the reason for the minor differences between the atlases. However, the overall pictures drawn by the atlases agree. Note also that we do not normalise the activations in this case, simply because they do not normalise the activations in the paper [9] introducing the Activation Atlas. However, the normalisation matters only for the averaging within grid-cells, not for the UMAP dimensionality reduction, since we use cosine-similarity as the similarity measure.

We perform this process on the layer 4c and obtain Figure 39. We can see from the Activation Atlas which concepts that lie close in the space of activations. Intuitively, we would hope that semantically similar concepts lie close in this space. The overall impression from Figure 39 is that concepts are grouped together in

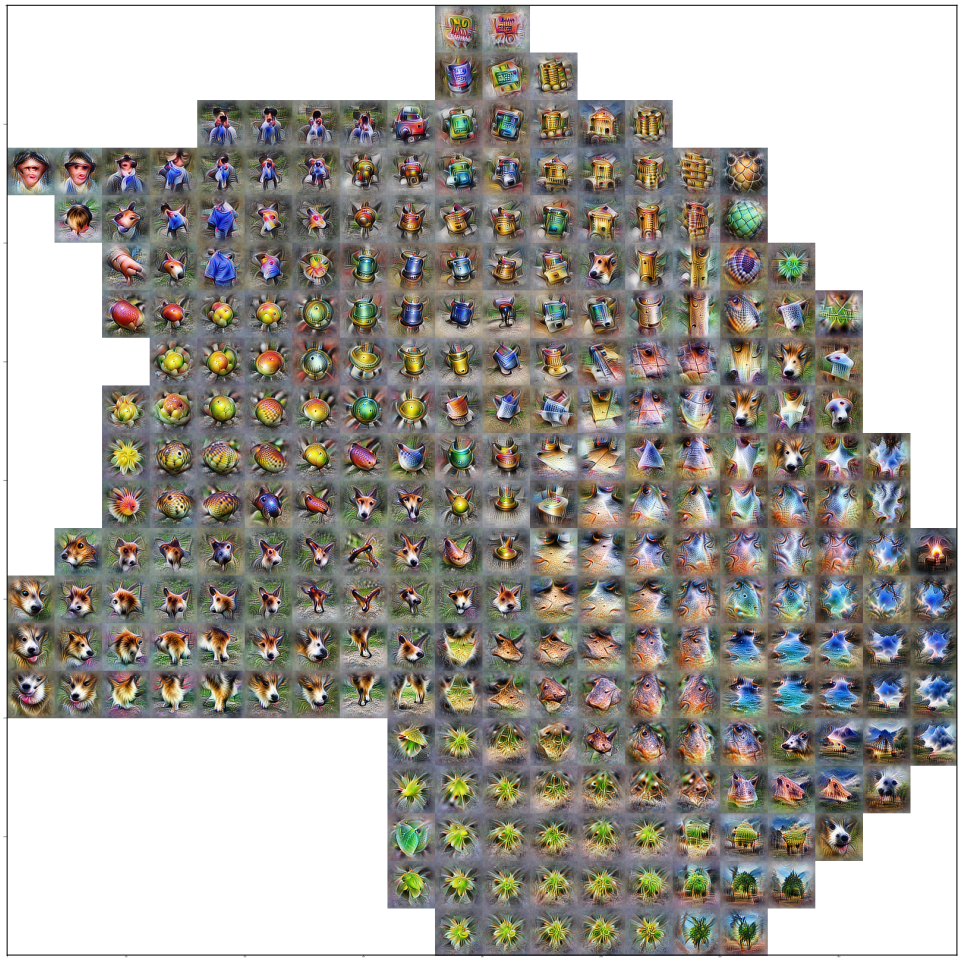


FIGURE 39. Activation Atlas for the 100 000 activations in layer $4c$, with grid size 20×20.

certain areas of the space, like animal features in the lower left flare, and humans and clothes in the upper left flare. Looking into the details, we can find small surprises, like the appearance of dog faces in the lower right and upper right regions. We might say that there seems to be some "semantic confusion". Why does this happen? Some possible explanations might be:

- Weaknesses in the feature visualisation method.
- The network actually associates semantically different concepts, that is, the Activation Atlas reveals the "true" story.
- The dimensionality reduction method fails to capture the actual shape of the space of activations.

Since feature visualisation is not the focus of this thesis, and because the feature visualisation methods seem to work quite well in general, we will not consider the first explanation.

If the Activation Atlas tells the true story (though not necessarily the full story), the semantic confusion could be valuable information about the network. It might tell us something about how the network is different from the human brain. Or perhaps the confusion is actually a weakness in the network, and that investigating it further could allow us to either improve it or exploit it. In general, the Activation Atlas could help us reveal things that look "weird" from the human perspective, and these things might be worthwhile to investigate further.

That the dimensionality reduction method fails to capture the actual shape of the space could again be due to different problems. We can roughly group them into two:

- Data problems: Not enough data, poor sampling methods, etc.
- Problems with the dimensionality reduction: The method is not appropriate or the task is unachievable.

For the first of these problems we refer to the discussion in the previous section on how we obtained the activations we are considering. The second set of problems might be resolved with a different method of dimensionality reduction, e.g. t-SNE instead of UMAP. But what if the space we are investigating simply cannot be sensibly embedded into $\mathbb{R}^2$? To overcome this, I propose to use Mapper instead of the dimensionality reduction method to create another type of Activation Atlas. Using Mapper, we are able to preserve more complex shapes from the high-dimensional space in our visualisation.

11.1. **Mapper Activation Atlas.** Let us describe the Mapper Activation Atlas pipeline. From the point cloud, e.g. P_{4c}, we construct a Mapper graph (with suitable parameters). In the Mapper graph, each node corresponds to a set of points from the original point cloud. We can therefore visualise each node as the feature visualisation of the average activation of the points belonging to the node. In this way we can capture the connectivity properties of the data in the high-dimensional space, without having to enforce the 2-dimensional embedding on it.

Let us see what a Mapper Activation Atlas can look like. Figure 40 shows a Mapper Activation Atlas of the point cloud P_{4c}. The images are a bit small for this format. In the same way that the Activation Atlas works better in an interactive format, like the format of the atlases presented in [9], the Mapper Activation Atlas would greatly benefit from an interactive format. Especially, zooming, hovering, rearrangement of the graph layout (which is somewhat arbitrary using `igraphs` [12] graph layout algorithms) and adjustment of the *min_intersection* parameter could greatly help to get more out of the Mapper Activation Atlas. All of these interactions could also be achieved without re-computation of the Mapper graph and visualisations of the nodes. Interactivity with parameters of the filter function, covering and clustering would of course be beneficial for choosing suitable parameters, but this would come at a significant computational cost. At least, the static figure presented here is a PDF, so zooming is encouraged.

It looks quite messy compared to the normal Activation Atlas in Figure 39. However, some features of the data seems to have become clearer in this atlas. For instance, in the lower right we see a flare consisting mainly of what seem to be dog faces, and this flare is quite weakly connected with the rest of the data. UMAP identifies this as a flare (lower left in Figure 39), but not as distinctly as Mapper does. In addition, the Mapper Activation Atlas seems to suggest that the part of

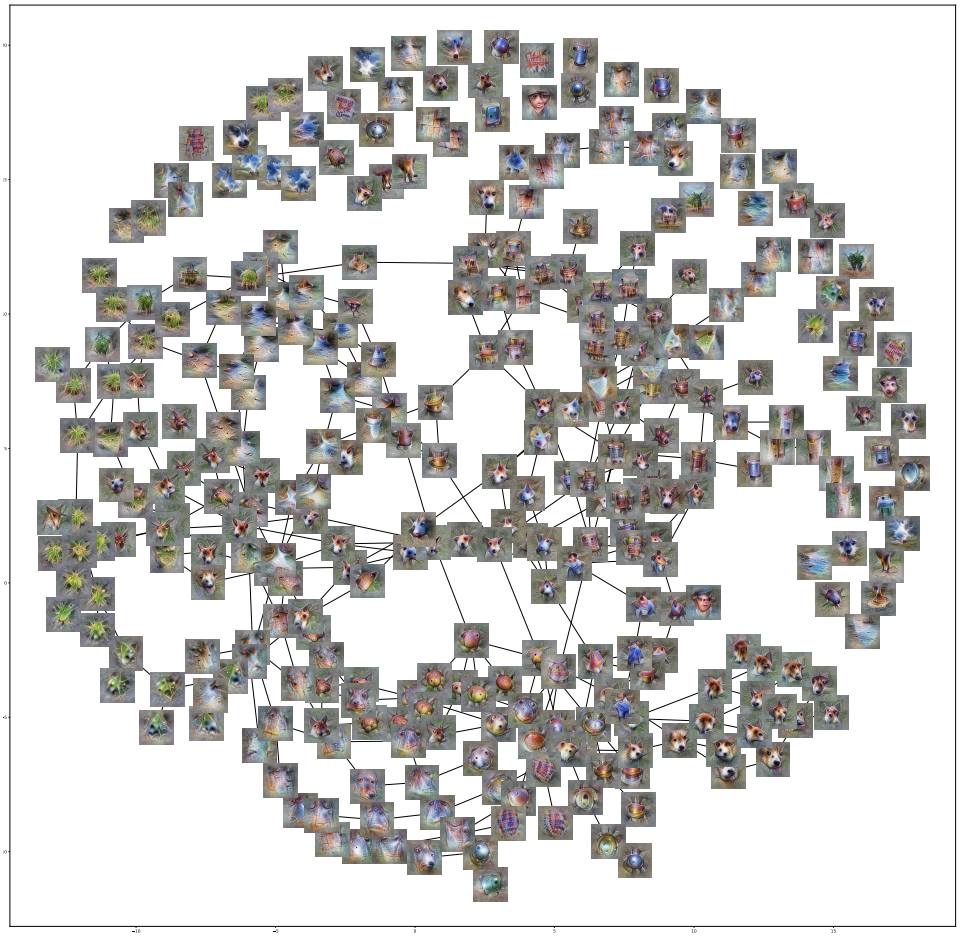


FIGURE 40. A Mapper Activation Atlas of P_{4c}. Projection on the first two principal components is used as the filter function. The cover is *balanced*, 8×8 rectangles with 0.3 overlap. The clustering used is k-means clustering with $k = 5$. The *min_intersection* is 100.

the space that is visualised as greenery is not as strongly connected to the rest of the space as the UMAP projection suggests. Lastly, also here we see the semantic confusion between animal faces and buildings, like that in the upper right region in Figure 39.

In Figure 41 we see a Mapper Activation Atlas of the same point cloud P_{4c}, but with fewer rectangles in the cover and with lower overlap. This creates a cleaner figure, while losing a lot of detailed information. It is encouraging to see that the dog-looking flare is still present in the lower right corner. This atlas seems to suggest that the main parts of the data roughly belong to one of five groups: greenery, buildings and human-made things, ground, fruit-like things and dog-like things. It also shows us how these concepts are connected. The presence of the loop with greenery in the left part of the figure illustrates a shortcoming of the original

FIGURE 41. A Mapper Activation Atlas of P_{4c}. Projection on the first two principal components is used as the filter function. The cover is *balanced*, 6×6 rectangles with 0.1 overlap. The clustering used is k-means clustering with $k = 5$. The *min_ intersection* is 40.

Activation Atlas: It does not preserve and visualise loops or holes well, in parts due to the dimensionality reduction and in parts due to grid structure enforced on the visualisations.

In a similar way as for the regular Activation Atlas, including *attribution* information into an interactive visualisation of the Mapper Activation Atlas could provide even more information about the workings of the CNN. This could for example allow us to understand which parts of the space that are typically relevant to specific classifications, as is also done with Activation Atlases in [9]. I further conjecture that attribution functions, either as measures of general importance or *class attributions*, could provide good filter functions for Mapper.

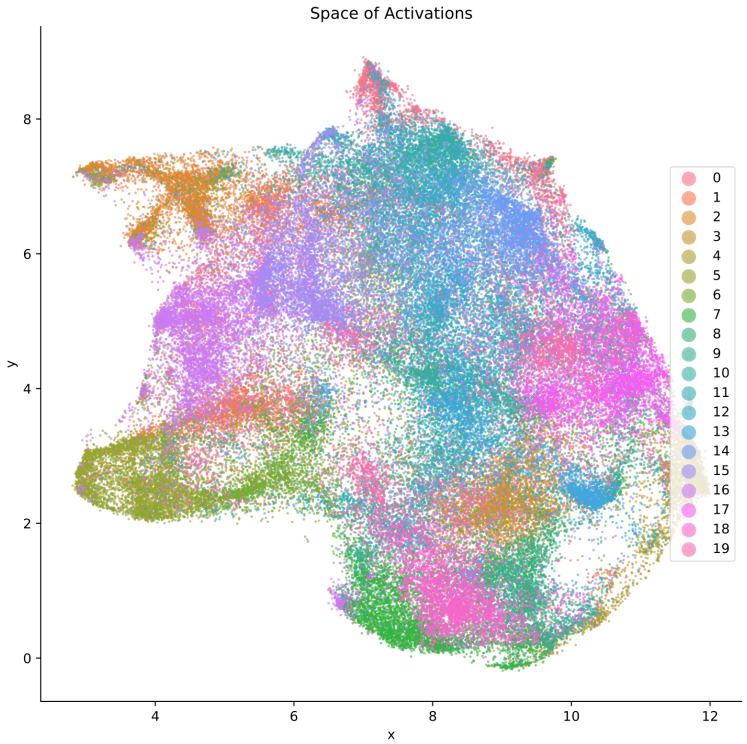


FIGURE 42. The whole space of activations P_{4c} visualised in 2d with UMAP. The colours correspond to clusters found with k-means clustering with $k = 20$.

FIGURE 43. Visualisations of cluster centroids of the clusters from Figure 42. The titles tell the cluster number and how many points belong to it. For example "c5 s5460" means that the visualisation is of cluster 5, and that cluster 5 contains 5260 directions.

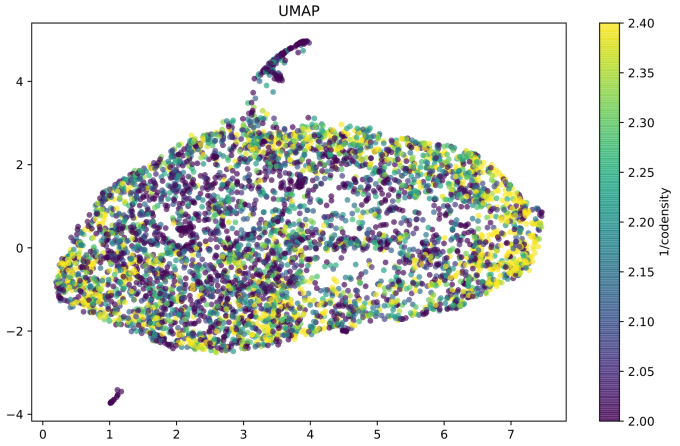


FIGURE 44. Visualisation with UMAP of the points from cluster 5 in Figure 42, coloured by 1/codensity. The axes are the two dimensions of the UMAP embedding of the data into two dimensions.

12. Finding Loops

In figure Figure 42 we visualise the data in two dimensions. This is a global view of the dataset. What happens if we zoom in, to look at the local structure in the dataset? While UMAP preserves some of the local structure, we can certainly preserve more by considering only the points in the region we are interested in. For points within a specific region it is also more likely that the uniformity assumption of UMAP is satisfied.

Let us investigate cluster number 5, which is illustrated in Figure 43 and located in the lower left of Figure 42. This cluster seems to correspond with the flare found in the Mapper figure Figure 40. We can start by visualising this subset of data, again using UMAP. See Figure 44. The high density region seems to form a loop in the data. Is this loop actually present in the data or an artefact of the dimensionality reduction? And if present in the data, what does it mean?

We can use persistent cohomology to investigate the presence of loops in the data. We do codensity filtration cf. Definition 6.13 with codensity parameter $k = 100$ to keep 2000 points and denoise cf. Definition 6.15 one iteration with denoising parameter $k = 100$. We then compute persistent cohomology of the resulting point cloud. The result is shown in Figure 45. We find a very persistent loop in the dataset. With circular parametrisation, we parametrise the loop found in the filtered dataset. Figure 45 shows UMAP visualisations of the data at different stages of preprocessing, coloured by the circular parameter. It is quite clear that the loop we find in the high-dimensional space is the same loop corresponding to the high-density regions in Figure 44. We confirm this in Figure 46 with similar plots, but with a PCA [22] dimensionality reduction. We conclude that the loop must be fairly flat, in the sense that it mainly lies in a plane. Moreover, the variance across the loop constitutes the main part of the variance in the dataset.

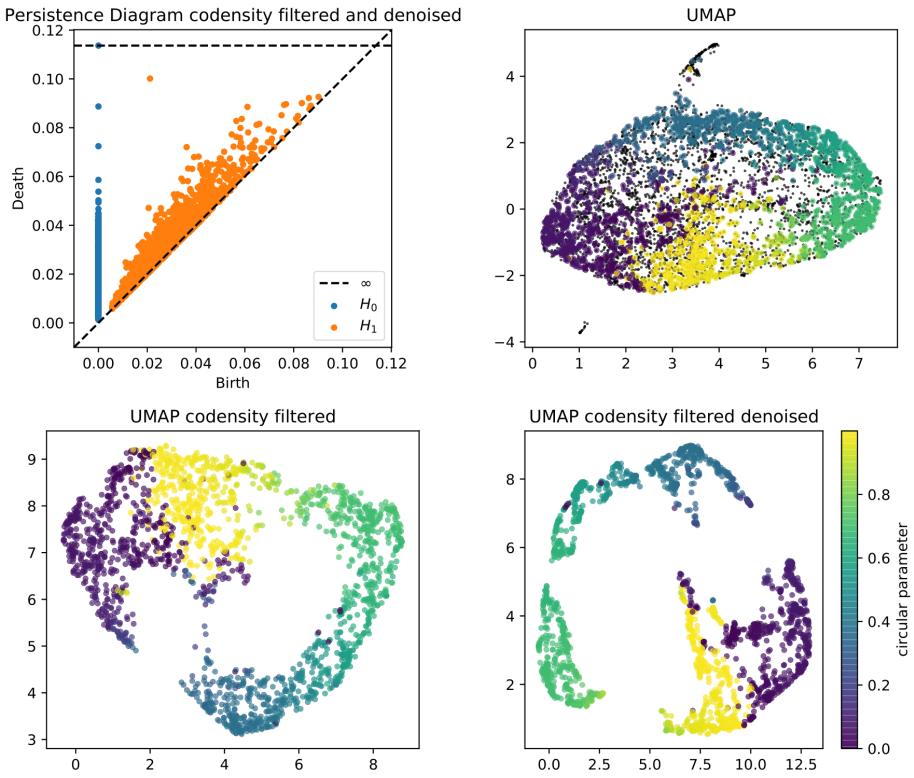


FIGURE 45. Persistence diagram for the points in cluster 5 from Figure 43, codensity filtered to 2000 points with codensity parameter $k = 100$, denoised one iteration with denoising parameter $k = 100$. UMAP visualisation of different datasets. Points are coloured by the circular parameter obtained for the most persistent cocycle found. Points that do not occur in the cocycle are coloured black.

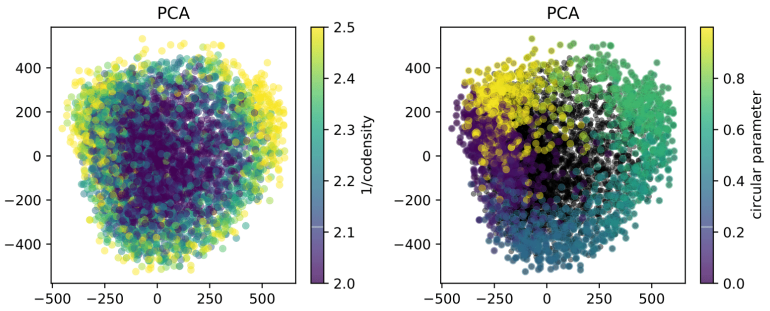


FIGURE 46. PCA projection of cluster 5 from Figure 43, coloured by inverse codensity and circular parameter. In the plot to the right, the points that are filtered out before computing the parametrisation are coloured black.

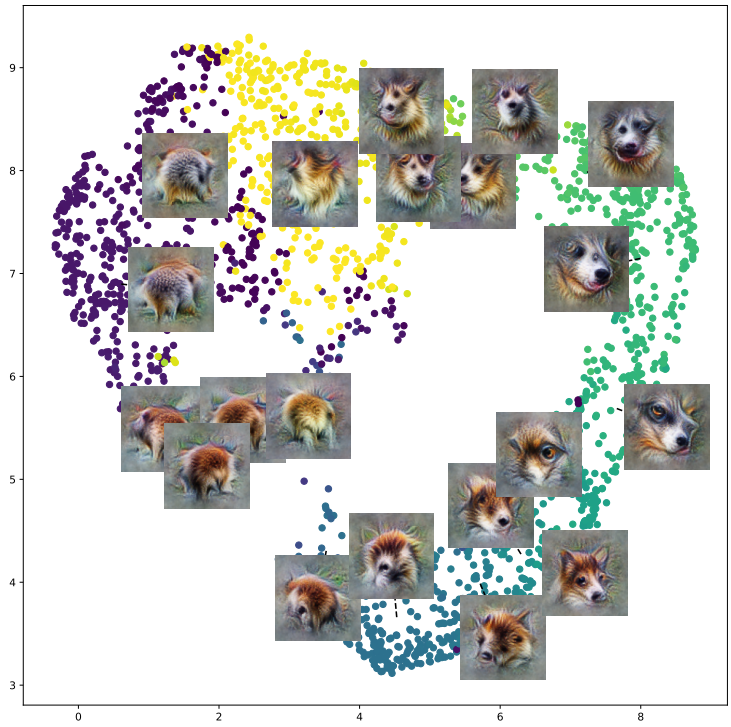


FIGURE 47. UMAP projection of codensity filtered data coloured by circular parameter. Visualisations of centroids of points binned according to the circular parameter value.

Remark 12.1. The parameters for the preprocessing are chosen through trial and error. We should therefore be careful not to draw premature conclusions based only on a persistence diagram obtained with a bit of "p-hacking" [58]. The other investigations that we carry out in addition to persistent homology justify the use of minor "p-hacking" to choose the parameters.

The next natural question to ask is: Why is there a loop in the cluster? Let us try to explore this by visualising the change that occurs along the circle. This is done in Figure 47.

From Figure 47 it seems like the points very roughly can be partitioned into 4 groups: Faces of small furry animals (bottom right), body and fur of small furry animals (bottom left), faces of larger animals with rougher fur (top right), and body and fur of larger animals with rougher fur (top left). Given this partition the circular structure makes sense. The connection between faces of small furry animals, and body parts of large animals with rough fur is not strong.

FIGURE 48. Mapper graph of cluster number 5 from Figure 43. First, the 2000 points of lowest codensity with codensity parameter $k = 100$ are filtered out. As filter functions we have used the projection on the first two principal components. The covering has 6 *balanced* intervals in each direction, with 0.3 overlap. For clustering we used k-means clustering with $k = 3$. We have *min_intersection* 5. The colour indicates the average value of the circular parameter within each node.

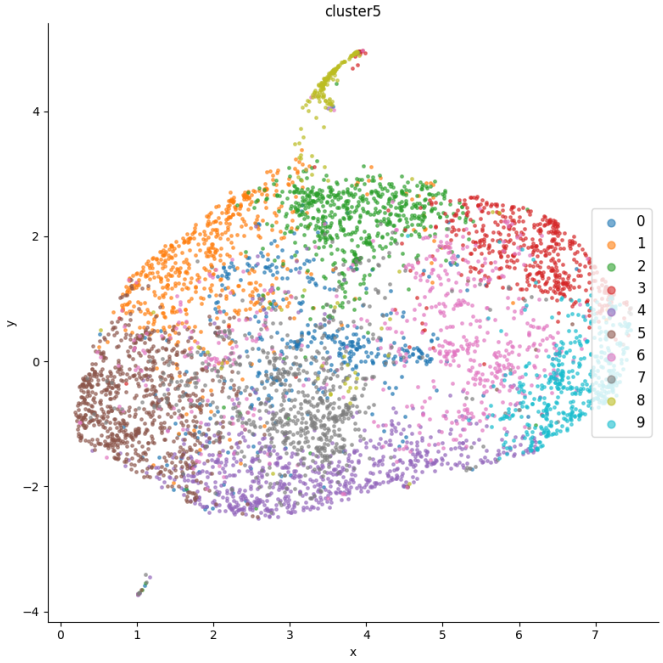


FIGURE 49. UMAP visualisation of the cluster, coloured with k-means clustering labels.

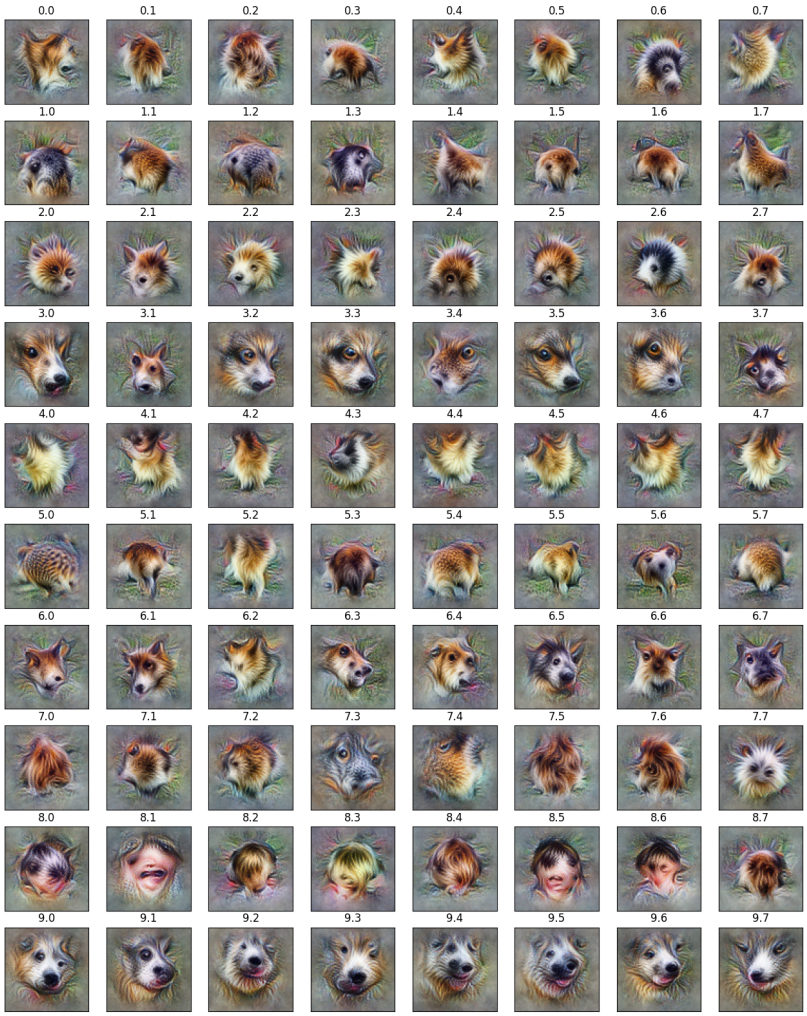


FIGURE 50. Visualisations of the clusters from Figure 49. For each cluster, we cluster again with k-means and visualise each cluster as the visualisation of a random representative from the cluster. The title $i.j$ means that the image is a visualisation of cluster j from cluster i from Figure 49.

We can reformulate the question we are asking to try to explain the loop in another way. Instead of trying to figure out what the loop "means", we can ask why we do not have (many) points in the hole the data loops around. Let us start by visualising the few points that lie in this hole. In Figure 49 we visualise the dataset where we colour by the clusters found with k-means clustering. Firstly, we notice that the dimensionality reduction seems to preserve most of the information found by k-means. From Figure 45 we see that mainly cluster 6 contains the low density points that lie in the hole we detected. In Figure 50 we visualise sample representatives from 8 clusters found with k-means clustering in each of the clusters

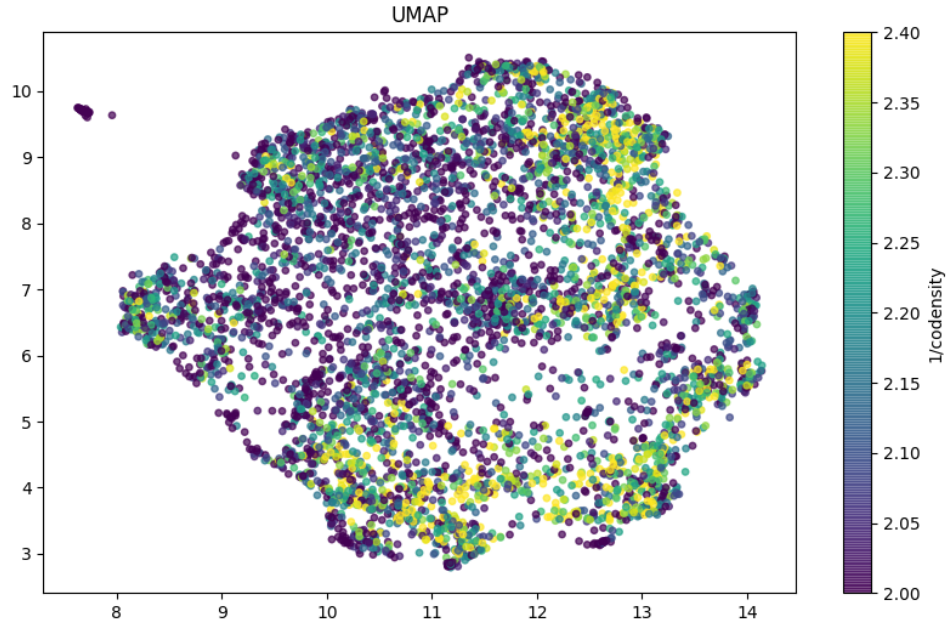


FIGURE 51. UMAP visualisation of the points belonging to flare 5 in layer 4b $P_{4b}^{(5)}$. Coloured by 1/codensity.

from Figure 49, e.g. the plot labelled 6.4 is a visualisation of a random representative from the fourth cluster within cluster number 6 from Figure 49. Thus each row contains visualisations that together should capture the variance within each cluster from Figure 49. Looking closely at the visualisations of cluster 6 (the row with titles $6.0, 6.1, \ldots, 6.7$), they look a bit like nonsensical mixes of faces of fairly different animals. Maybe this can explain the low density in the hole. It seems at least that the other clusters represent slightly clearer and more understandable concepts.

12.1. **Same Cluster in Other Layers.** Each image in the input space corresponds to a vector in P_{4c}, but also to a vector in P_{4b} and P_{4d}. Thus, we get a bijection between the points in P_{4c} and those in P_{4b}, or any other layer. Using this, we might ask the question: What does the loop we have just found look like in the other layers?

Let $P_{4c}^{(5)} \subseteq P_{4c}$ denote the points in cluster number 5 from Figure 43 that we just studied. By $P_{4b}^{(5)} \subseteq P_{4b}$ we refer to the points in P_{4b} that correspond to $P_{4c}^{(5)}$ under the bijection just introduced. We similarly define $P_{4d}^{(5)} \subseteq P_{4d}$ and the subset in the input space $P^{(5)} \subseteq P$. We let $Q_{4c}^{(5)} \subseteq P_{4c}^{(5)}$ denote the codensity filtered cluster with 2000 points and codensity parameter $k = 100$, as in Figure 45. Again we define $Q_{4b}^{(5)}$, $Q_{4d}^{(5)}$ and $Q^{(5)}$ via the bijection. Thus, $Q_{4b}^{(5)}$ refer to the codensity filtered cluster, but the codensity is calculated from the distances in the space P_{4c}, not in P_{4b}.

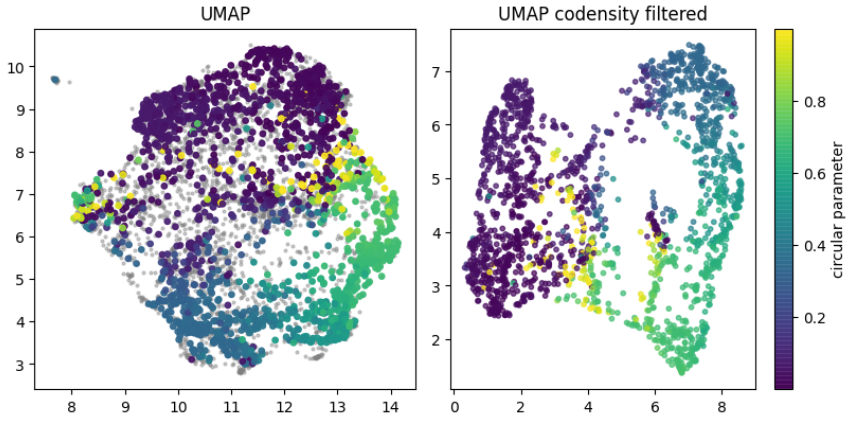


FIGURE 52. UMAP visualisations of the points in the flare $P_{4b}^{(5)}$ (left) and codensity filtered $Q_{4b}^{(5)}$ (right). Points that do not appear in the cocycle are coloured gray.

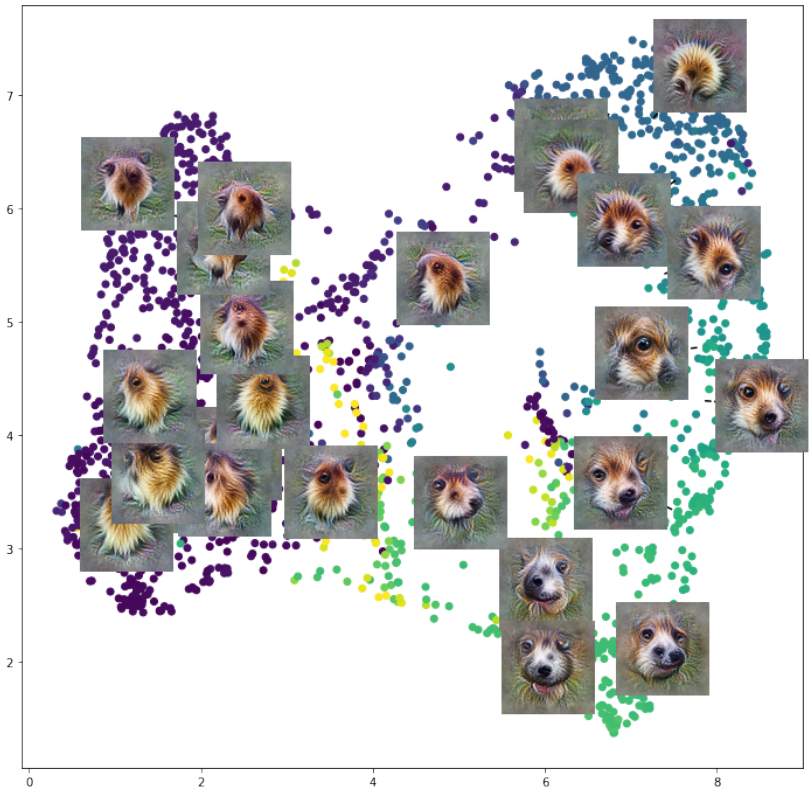


FIGURE 53. UMAP projection of codensity filtered data $Q_{4b}^{(5)}$ coloured by circular parameter found in $Q_{4c}^{(5)}$. Visualisations of centroids of points binned according to the circular parameter value.

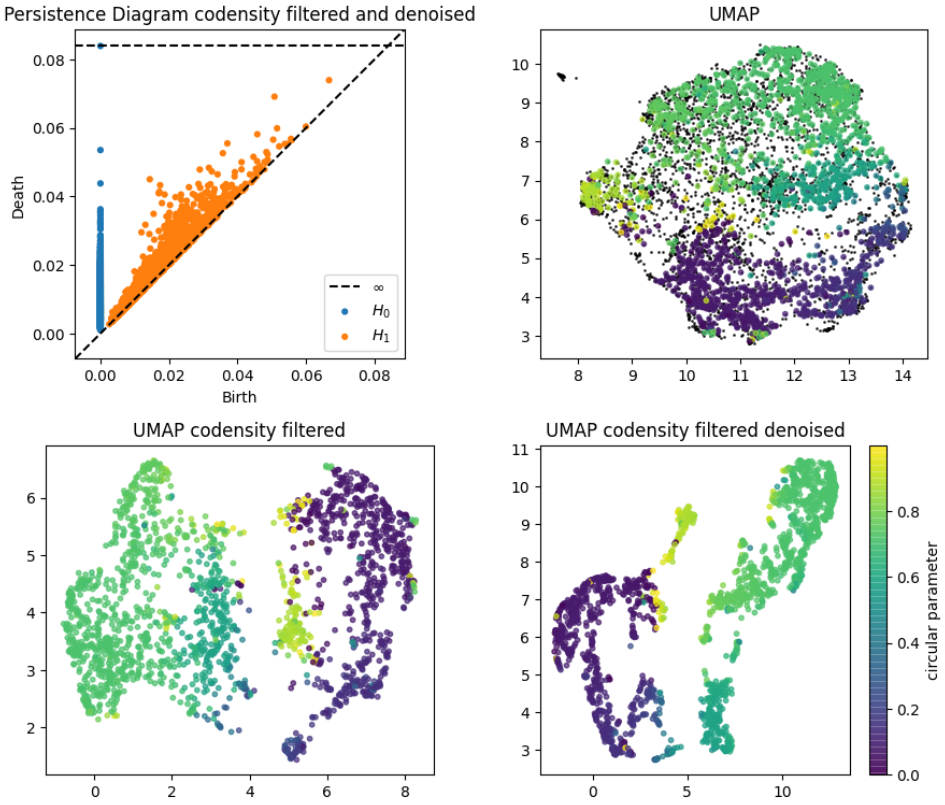


FIGURE 54. Persistence diagram for the points in $P^{(5)}_{4b}$, codensity filtered to 2000 points with codensity parameter $k = 200$, denoised one iteration with denoising parameter $k = 300$. UMAP visualisation of different datasets. Points are coloured by the circular parameter obtained for the most persistent cocycle found. Points that do not occur the cocycle are coloured black.

12.1.1. *Layer 4b.* We start by visualising $P^{(5)}_{4b}$ in Figure 51. There seems to be a hole in this space as well, but it is definitely not the main feature of the space, as it is in layer 4c. Is the hole we can see in Figure 51 the same as that in $P^{(5)}_{4c}$? Via the bijection we can also associate the circular parameter we computed on $Q^{(5)}_{4c}$ to the points in $Q^{(5)}_{4b}$. We colour the points with this parameter in Figure 52, to see if the loops correspond.

It seems that parts of the loop we see in Figure 45 corresponds to the one we see in Figure 51, since the circular parameter found in $Q^{(5)}_{4c}$ wraps around the whole in $Q^{(5)}_{4b}$. The image is, however, not as clean in layer 4b as in 4c. For example, there is a green area in the left of the left plot in Figure 51, in addition to some general disagreement between the visualised loop and the circular parameter in both plots. This might be the explanation for the comparably high difficulty of detecting loops

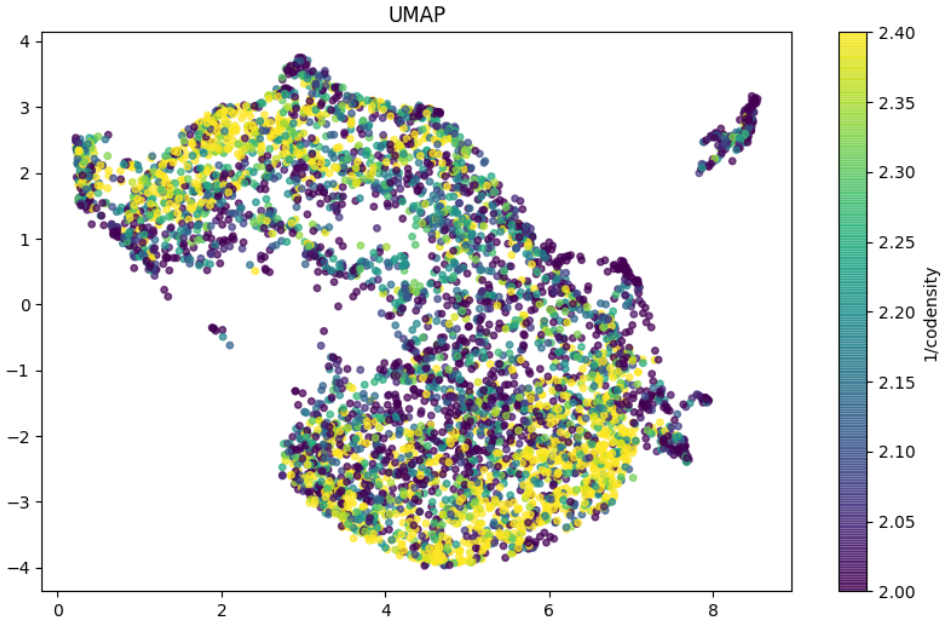


FIGURE 55. UMAP visualisation of the points belonging to flare 5 $P^{(5)}_{4d}$ in layer 4d. Coloured by 1/codensity.

in $Q^{(5)}_{4b}$ with persistent homology. It is, however, clear that the circular parameter found in $Q^{(5)}_{4c}$ to a large extent explains the loops we see in Figure 52.

In Figure 53 we visualise the points along the loop detected in $Q^{(5)}_{4b}$. The visualisations tell a similar story to that in Figure 47, but it seems the concepts are not as clearly separated by the CNN in this layer.

Figure 54 illustrates my most successful attempt at a circular parametrisation of the space $P^{(5)}_{4b}$, with codensity filtration, denoising and persistent cohomology all computed from the distances in P_{4b}. The circular parameter does seem to describe the loop around the lower right hole, but the loop is clearly not as dominant as in $P^{(5)}_{4c}$, cf. Figure 45. An example with a poor parametrisation is of course not any proof that no good parametrisation can be found, but Figure 54 is illustrative of the results obtained by choosing a number (~ 10) of different parameters for the preprocessing. This is quite contrary to the loop found in $P^{(5)}_{4c}$, which was distinctly detected by persistent homology for most choices of reasonable preprocessing parameters.

12.1.2. *Layer 4d.* We can also investigate what happens with the flare in the next layer, that is we investigate $P^{(5)}_{4d}$. As always, we start with a simple UMAP visualisation, coloured by 1/codensity in Figure 55. Here, it seems like the the loop has been cut, and is in the process of splitting into two distinct components. This is confirmed also by the clustering of the whole space of activations in layer $4d$, shown in Figure 56. Figure 57 shows the overlaps of each cluster obtained with k-means

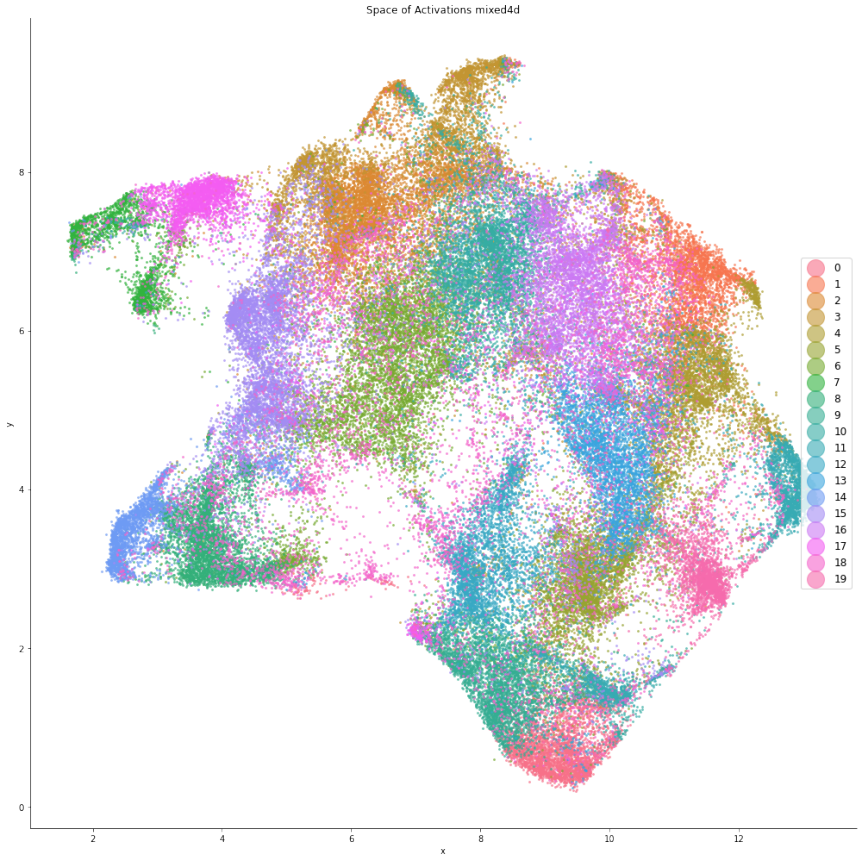


FIGURE 56. The whole space of activations P_{4d} visualised with UMAP. The colours correspond to clusters found with k-means clustering with $k = 20$.

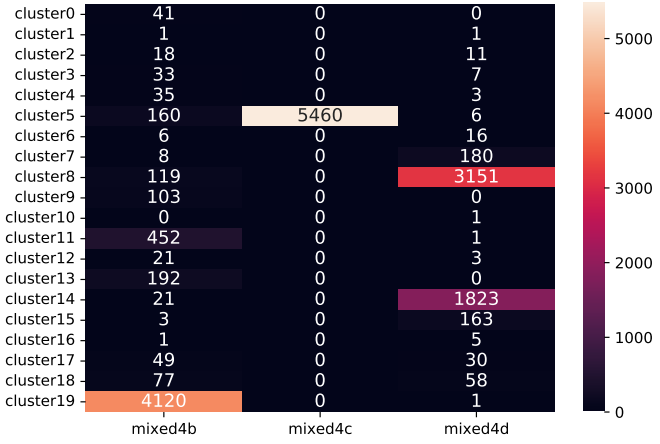


FIGURE 57. Number of common points with $P_{4c}^{(5)}$ for each cluster found with k-means on the point clouds P_{4b} and P_{4d}.

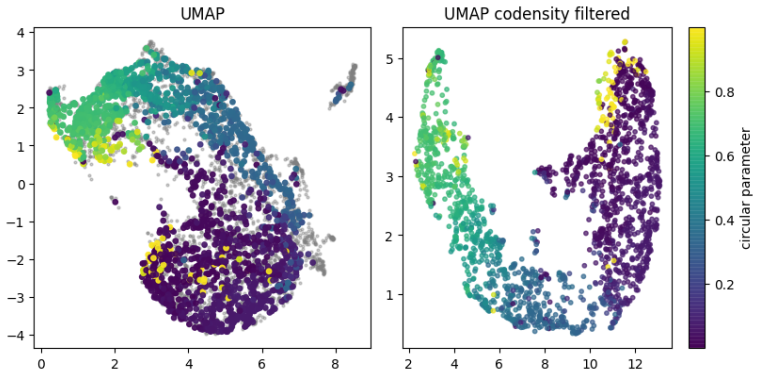


FIGURE 58. UMAP visualisations of the points in the flare $P^{(5)}_{4d}$ (left) and codensity filtered $Q^{(5)}_{4d}$. Points that do not appear in the cocycle are coloured gray.

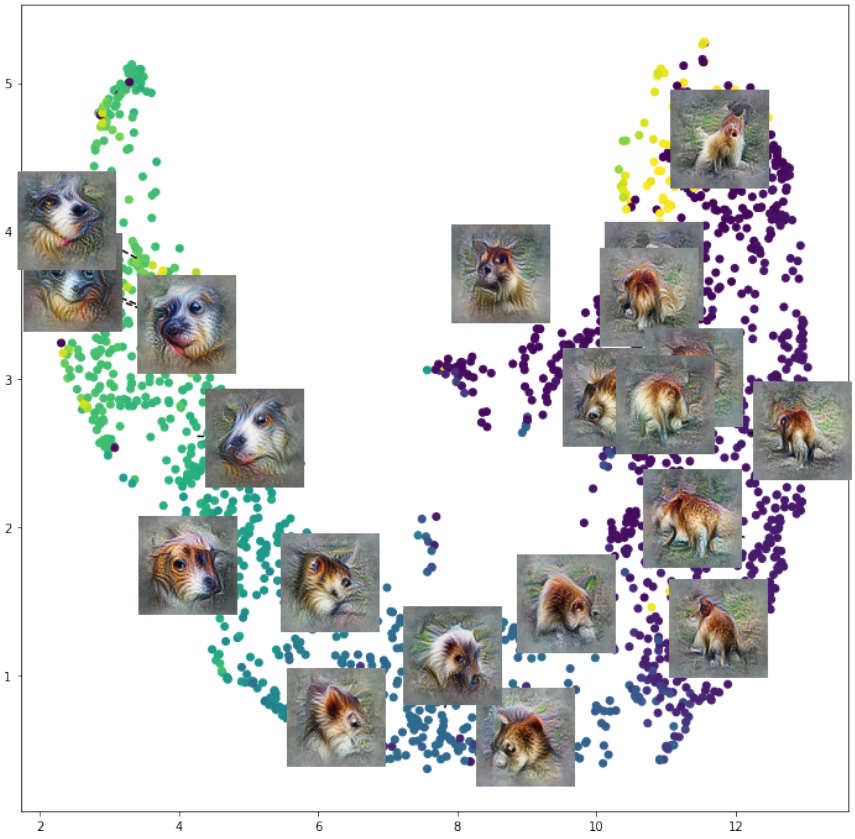


FIGURE 59. UMAP projection of codensity filtered data $Q^{(5)}_{4d}$ coloured by circular parameter found in $Q^{(5)}_{4c}$. Visualisations of centroids of points binned according to the circular parameter value.

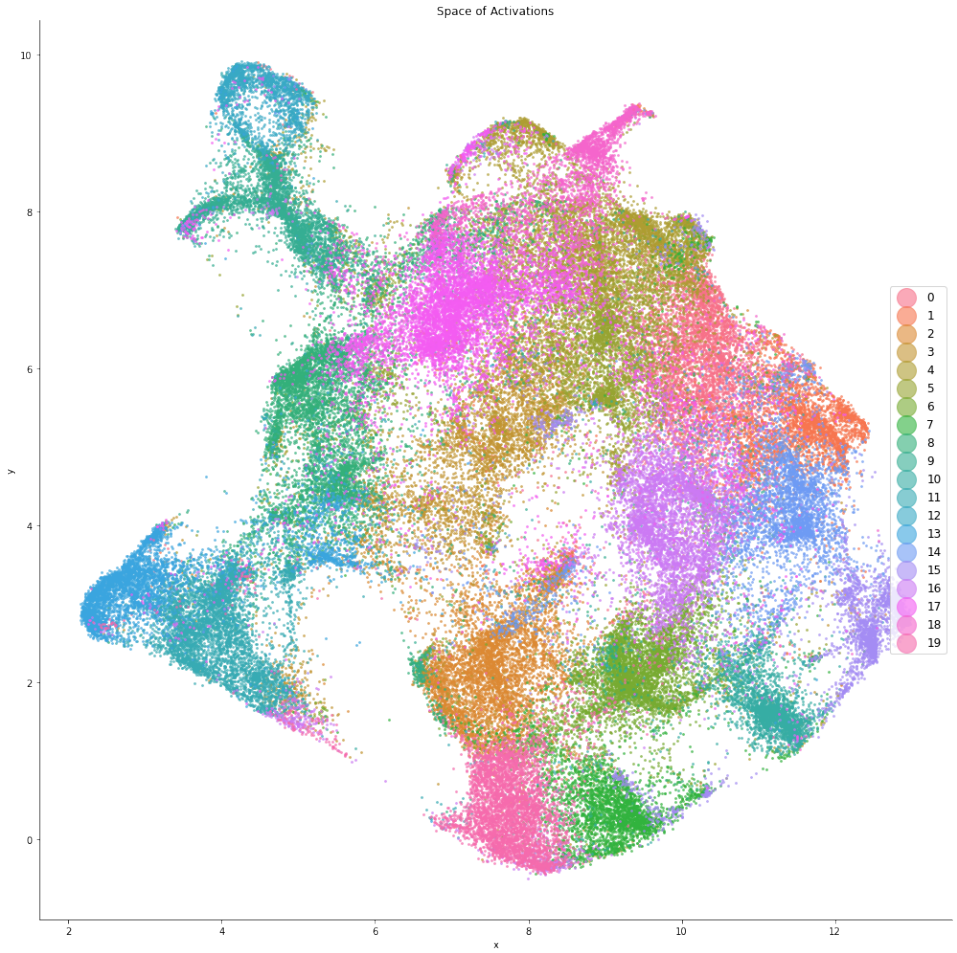


FIGURE 60. The whole space of activations P_{4e} visualised in 2d with UMAP. The colours correspond to clusters found with K-means clustering with $k = 20$.

clustering on P_{4b} and P_{4d}. We see that in $4b$ the cluster $P_{4c}^{(5)}$ is almost entirely contained within a cluster. In $4d$, the cluster $P_{4c}^{(5)}$ has split into two different clusters. We can also notice from Figure 56 that the flare we are investigating (bottom left corner) seems to become more separated from the rest of the full space.

Figure 58 confirms that the loop that has been cut apart is the same loop that we found in $P_{4c}^{(5)}$. Maybe the loop we found was simply a symptom of the way that layers in the CNN disentangle concepts. The network seems to tear a hole in the middle of the "disk" and then tear it apart into two distinct semi-circles. A question to investigate further could be: Is this a common way for the network to divide a "disk" into two parts, and why does it happen (at least here)?

In Figure 59 we see that the two flares that are cut apart do not have much in common anymore. It seems like the network at this point starts to make a large distinction between animal faces and furry body parts.

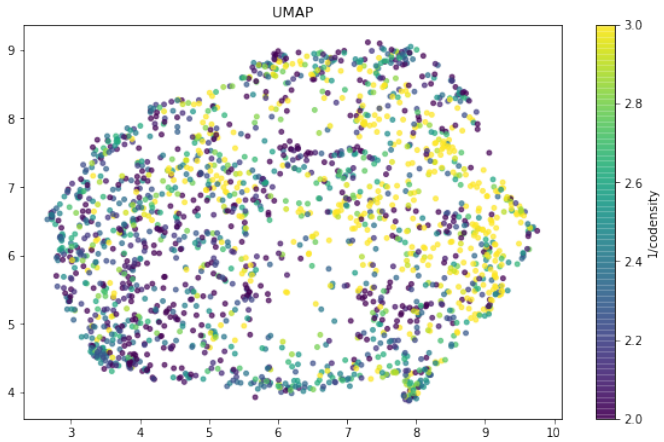


FIGURE 61. Visualisation with UMAP of the points from cluster 5 in Figure 42, coloured by 1/codensity.

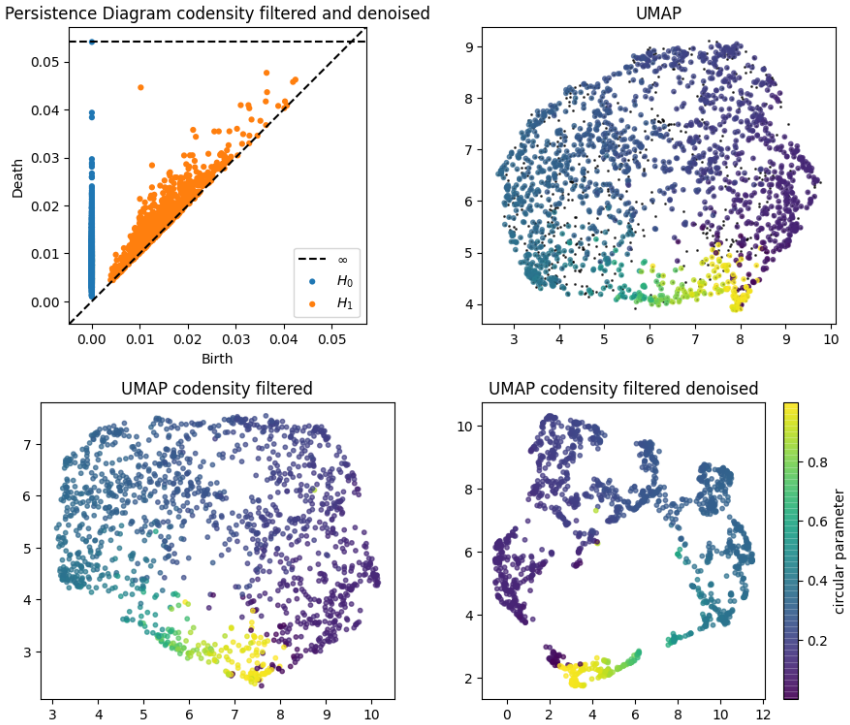


FIGURE 62. Persistence diagram for the points in cluster 12 from Figure 60, codensity filtered to 1500 points with codensity parameter $k = 100$, denoised one iteration with denoising parameter $k = 100$. UMAP visualisation of different datasets. Points are coloured by the circular parameter obtained for the most persistent cocycle found. Points that do not occur in the cocycle are coloured black.

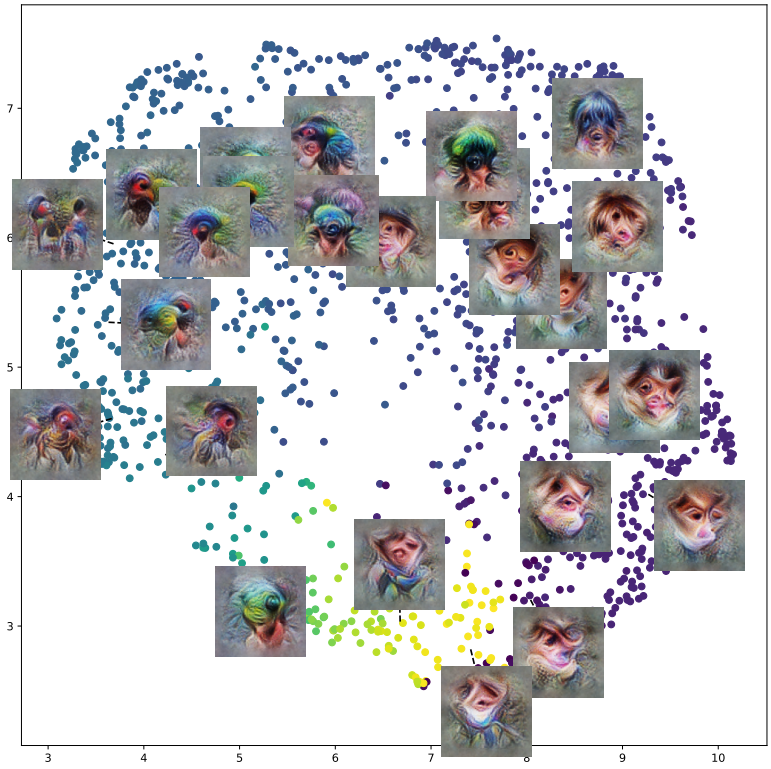


FIGURE 63. UMAP projection of codensity filtered data in cluster 12 from Figure 60, coloured by circular parameter. Visualisations of centroids of points binned according to the circular parameter value.

12.2. **Cluster in Layer 4e.** Let us briefly review another example of a loop found using a similar approach. We first start by visualising the full space of activations P_{4e} in Figure 60.

We see that cluster number 12 in the upper left corner is quite weakly connected from the rest and seems to exhibit a loop structure according to the UMAP dimension reduction. The visualisation of the cluster in Figure 61 shows mild symptoms of a loop. After codensity filtration and denoising, persistent cohomology detects a very significant loop in the data, and the circular parameter is intuitive, as illustrated in Figure 62.

Figure 63 displays a visualisation of the data points along the circular parameter. We will not go further into the details to try to justify the presence of this loop. The example is meant to illustrate how the method is able to identify loops in the data rather easily. Similar investigations have been performed on all the other clusters in P_{4c} from Figure 42. In no clusters, except cluster 5, did the UMAP visualisation as in Figure 42 reveal a large circular structure. Persistent cohomology of codensity filtered and denoised clusters rarely gave significant features. In the rare cases when it did, the circular parameter, visualised as in Figure 45 seemed nonsensical, suggesting that the loops detected were of the spurious, or at least local, sort.

13. Further Work

In this final section I will summarise and present ideas for further work that seem worthwhile pursuing, based on the intuition achieved through the work on this thesis. Some of the ideas have already been discussed or presented in the relevant sections, while some are first introduced in this section.

13.1. **Collection of Activations.** In Section 10, the method used to obtain channel activation vectors, and their normalised directions, is explained and discussed. I also propose another way of collecting activation vectors, randomly choosing patches from images and averaging across each of the channel activations. Whether the point clouds obtained in such a way would be similar or not to the point clouds we have investigated in this thesis is difficult to say. However, regardless of which point cloud that is the most "suitable", investigation of the agreements and disagreements seem interesting and potentially enlightening to me.

13.2. **Attribution.** As briefly mentioned in Section 10, good attribution functions could provide useful additional information about the point clouds of activations that we are investigating. If we had a measure of *importance* the network associates with an activation, we could use this to focus our attention on the activations of high importance, at least initially.

A consequence of both sampling methods discussed previously, is that the distribution of the activations reflect the distribution of the occurrence of patches (or concepts) in the images. For example, activations associated with grass are extremely common, simply since they are very common in images. These activations are not very relevant to the classifications of images. On the other hand, human features are very relevant for classifications, but are not very common in the images. For some purposes this could be unfortunate. For example, it makes (co)density filtrations across very different areas of the point clouds a bad idea. For dimensionality reduction methods, like UMAP for the Activation Atlas, the dense regions containing things like grass, ground and water, might be given too much attention, at the cost of high-attribution regions. Binning, filtering or re-sampling based on attribution values seem like good ideas.

13.3. **Layers.** In Section 12 we investigate the same subset of activations across multiple layers, to see the evolution of the loop found in layer $4c$. This is a very basic example of the use of the complex structure of the dataset that is due to the correspondence between activations in different layers.

Given the random sampling of a spatial activation, does the correspondence between layers constitute a sensible (whatever that means) function between the spaces we have investigated? It is reasonable to expect a spatial activation with a visualisation like a dog face in P_{4c}, to not have a visualisation like water in P_{4d}. On the other hand, if the functions between layers did "nothing", why would we need so many layers, which we know that we do? We can therefore expect the functions between layers to do more than "nothing", while still preserving some properties of the space. One might expect that we could design sensible continuous functions between simplicial complexes associated with the point clouds from the correspondences. Given such functions, the connection between "doing nothing" and the functions being homeomorphisms is intuitive. This vague observation together

with the *functoriality* of most topological methods suggest that topological methods might be suitable for the study of the transformations between layers.

13.4. **Compare Different CNNs.** To compare the point clouds of activations across different CNNs seems like a fascinating idea. To what extent could one find agreements between the networks? In the case of disagreements: Are any of the behaviours preferable? Can we associate the disagreements with any known strengths or weaknesses of the different models? Explicit questions could be: Does a vgg16 CNN [49] have a layer where the activations look similar to P_{4c}, with the "dog flare"? Can we find the same loops in the two networks? Do the networks show the same kinds of "semantic chaos" in (Mapper) Activation Atlases?

13.5. **Interactive Mapper Activation Atlas.** Methods like the Activation Atlas benefit greatly from being interactive, since this allows for more information to be presented in an approachable way. Moreover, interactivity can give robustness and better understanding regarding hyperparameters. As discussed in Section 11, I believe the Mapper Activation Atlas would benefit greatly from an interactive implementation, especially with regards to the *min_ intersection* parameter, zooming, and manual rearrangement of the graph layout.

13.6. **Higher Resolution Mapper.** In Section 11, we introduced the Mapper Activation Atlas. In the pipeline described to create the Mapper Activation Atlas, the number of covers in the cover used to compute the Mapper graph, determines the number of nodes in the Mapper graph and thus the number of images, i.e. the resolution, of the Mapper Activation Atlas. Hence, the "dimensionality reduction", that is, the creation of the graph from the data, depends on the resolution we want for the output visualisation. For example, we might want to compute the Mapper graph from the data with a very high number of covers, thus obtaining a high number of nodes in the graph. With the simple pipeline proposed in Section 11, the resulting Mapper Activation Atlas would have a very high number of nodes, and therefore too many visualisations of directions to be of any use whatsoever. This might be unfortunate.

I propose to separate the resolution of the Mapper graph and the resolution of the Mapper Activation Atlas visualisation. A simple approach would be to visualise only a subset of the nodes in the Mapper graph, based on the intuition that points that are closely connected probably have very similar visualisations. Other approaches could involve some sort of "clustering" of the nodes in the Mapper graph, or binning of nodes based on some functions, such as a circular parameter. Regardless of the approach, the goal is to allow computation of a Mapper graph with a high number of nodes for a potentially more accurate representation of the shape of the space, while still making the Mapper Activation Atlas visually approachable for humans.

Finally, it is worth mentioning that the Mapper Activation Atlas could certainly be improved by making better choices of methods and parameters, especially for the clustering method.

References

[1] Louis Abraham. *tomaster: Topological Mode Analysis on Steroids*. https://pypi.org/project/tomaster/. 2021.

[2] Henry Adams and Michael Moy. *Topology Applied to Machine Learning: From Global to Local*. 2021. arXiv: 2103.05796 [math.AT].

[3] Charu C. Aggarwal. *Neural Networks and Deep Learning. A Textbook*. Cham: Springer, 2018, p. 497. ISBN: 978-3-319-94462-3. DOI: 10.1007/978-3-319-94463-0.

[4] Ulrich Bauer. "Ripser: efficient computation of Vietoris-Rips persistence barcodes". In: *J. Appl. Comput. Topol.* 5.3 (2021), pp. 391–423. ISSN: 2367-1726. DOI: 10.1007/s41468-021-00071-5. URL: https://doi.org/10.1007/s41468-021-00071-5.

[5] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[6] Gunnar Carlsson. "Topology and data". In: *Bull. Amer. Math. Soc. (N.S.)* 46.2 (2009), pp. 255–308. ISSN: 0273-0979. DOI: 10.1090/S0273-0979-09-01249-X. URL: https://doi.org/10.1090/S0273-0979-09-01249-X.

[7] Gunnar Carlsson and Rickard Brüel Gabrielsson. "Topological Approaches to Deep Learning". In: *Topological Data Analysis*. Ed. by Nils A. Baas et al. Cham: Springer International Publishing, 2020, pp. 119–146. ISBN: 978-3-030-43408-3.

[8] Gunnar Carlsson et al. "On the local behavior of spaces of natural images". In: *Int. J. Comput. Vis.* 76.1 (2008), pp. 1–12. ISSN: 0920-5691. DOI: 10.1007/s11263-007-0056-x. URL: https://doi.org/10.1007/s11263-007-0056-x.

[9] Shan Carter et al. "Activation Atlas". In: *Distill* (2019). https://distill.pub/2019/activation-atlas. DOI: 10.23915/distill.00015.

[10] Frédéric Chazal et al. "Persistence-Based Clustering in Riemannian Manifolds". In: *Journal of the ACM* 60 (June 2011). DOI: 10.1145/1998196.1998212.

[11] Chao Chen et al. "TopoReg: A Topological Regularizer for Classifiers". In: *CoRR* abs/1806.10714 (2018). arXiv: 1806.10714. URL: http://arxiv.org/abs/1806.10714.

[12] Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research". In: *InterJournal* Complex Systems (2006), p. 1695. URL: https://igraph.org.

[13] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[14] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010, pp. I–XII, 1–241. ISBN: 978-0-8218-4925-5.

[15] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. "Topological Persistence and Simplification". In: (Jan. 2003).

[16] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[17] Hitesh Gakhar and Jose A. Perea. *Sliding Window Persistence of Quasiperiodic Functions*. 2021. arXiv: 2103.04540 [math.AT].

[18] Ingo Gühring, Mones Raslan, and Gitta Kutyniok. *Expressivity of Deep Neural Networks*. 2020. arXiv: 2007.04759 [cs.LG].

[19] Allen Hatcher. *Algebraic topology*. Cambridge: Cambridge University Press, 2002, pp. xii+544. ISBN: 0-521-79160-X; 0-521-79540-0.

[20] Rune Haugseng. *Lecture notes in Algebraic Topology 1*. Nov. 2020.

[21] Felix Hensel, Michael Moor, and Bastian Rieck. "A Survey of Topological Machine Learning Methods". In: *Frontiers in Artificial Intelligence* 4 (2021), p. 52. ISSN: 2624-8212. DOI: 10.3389/frai.2021.681108. URL: https://www.frontiersin.org/article/10.3389/frai.2021.681108.

[22] Harold Hotelling. "Analysis of a complex of statistical variables into principal components." In: *Journal of Educational Psychology* 24 (1933), pp. 498–520.

[23] John M Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596 (2021), pp. 583–589.

[24] J. B. Kruskal. "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis". In: *Psychometrika* 29 (1964), pp. 1–27. ISSN: 0033-3123. DOI: 10.1007/BF02289565. URL: https://doi.org/10.1007/BF02289565.

[25] Leland McInnes and Tim Sainburg and Graham Clendenning and Sebastian Pujalte. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. https://umap-learn.readthedocs.io/en/latest/index.html. [Online; accessed 20-December-2021]. 2021.

[26] Lei Li et al. "Identification of type 2 diabetes subgroups through topological analysis of patient similarity". In: *Science Translational Medicine* 7 (2015), 311ra174–311ra174.

[27] Ephy Love et al. "Topological Deep Learning". In: *ArXiv* abs/2101.05778 (2021).

[28] Pek Yee Lum et al. "Extracting insights from the shape of complex data using topology". In: *Scientific Reports* 3 (2013).

[29] Laurens van der Maaten and Geoffrey E. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[30] Leland McInnes and John Healy. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: (Feb. 2018).

[31] Michael Moor et al. *Topological Autoencoders*. 2021. arXiv: 1906.00722 [cs.LG].

[32] J.R. Munkres. *Topology*. Featured Titles for Topology. Prentice Hall, Incorporated, 2000. ISBN: 9780131816299. URL: https://books.google.no/books?id=XjoZAQAAIAAJ.

[33] James R. Munkres. *Elements of Algebraic Topology*. Addison Wesley Publishing Company, 1984. ISBN: 0201045869. URL: http://www.worldcat.org/isbn/0201045869.

[34] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: *Distill* (2017). https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

[35] Chris Olah et al. "The Building Blocks of Interpretability". In: *Distill* (2018). https://distill.pub/2018/building-blocks. DOI: 10.23915/distill.00010.

[36] Steve Y. Oudot. *Persistence Theory: From Quiver Representations to Data Analysis*. Mathematical Surveys and Monographs 209. American Mathematical Society, 2015, p. 218. URL: https://hal.inria.fr/hal-01247501.

[37] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[38] Jose A. Perea. *A Brief History of Persistence*. 2019.

[39] Jose A. Perea and John Harer. "Sliding windows and persistence: an application of topological methods to signal analysis". In: *Found. Comput. Math.* 15.3 (2015), pp. 799–838. ISSN: 1615-3375. DOI: 10.1007/s10208-014-9206-z. URL: https://doi.org/10.1007/s10208-014-9206-z.

[40] Birk Ramberg. *An Introduction to Persistent Homology*. June 2021.

[41] Bastian Rieck et al. "Neural Persistence: A Complexity Measure for Deep Neural Networks Using Algebraic Topology". In: *CoRR* abs/1812.09764 (2018). arXiv: 1812.09764. URL: http://arxiv.org/abs/1812.09764.

[42] Olga Russakovsky et al. "ImageNet large scale visual recognition challenge". In: *Int. J. Comput. Vis.* 115.3 (2015), pp. 211–252. URL: https://doi.org/10.1007/s11263-015-0816-y.

[43] Erik Rybakken, Nils Baas, and Benjamin Dunn. "Decoding of neural data using cohomological feature extraction". In: *Neural Comput.* 31.1 (2019), pp. 68–93. ISSN: 0899-7667. DOI: 10.1162/neco_a_01150. URL: https://doi.org/10.1162/neco_a_01150.

[44] Erich Schubert et al. "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN". In: *ACM Trans. Database Syst.* 42.3 (2017), Art. 19, 21. ISSN: 0362-5915. DOI: 10.1145/3068335. URL: https://doi.org/10.1145/3068335.

[45] Ludvig Schubert. *Lucid*. https://github.com/tensorflow/lucid. 2021.

[46] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization". In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://dx.doi.org/10.1007/s11263-019-01228-7.

[47] Vin de Silva and Mikael Vejdemo-Johansson. "Persistent cohomology and circular coordinates". In: *Proceedings of the 25th annual symposium on Computational geometry - SCG '09* (2009). DOI: 10.1145/1542362.1542406. URL: http://dx.doi.org/10.1145/1542362.1542406.

[48] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps." In: *CoRR* abs/1312.6034 (2013). URL: http://dblp.uni-trier.de/db/journals/corr/corr1312.html#SimonyanVZ13.

[49] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2015).

[50] Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. "Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition". In: *Eurographics Symposium on Point-Based Graphics*. Ed. by M. Botsch et al. The Eurographics Association, 2007. ISBN: 978-3-905673-51-7. DOI: 10.2312/SPBG/SPBG07/091-100.

[51] David I. Spivak. *Metric Realization of Fuzzy Simplicial Sets*. 2012.

[52] Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Online; accessed 20-December-2021]. 2018.

[53] Christian Szegedy et al. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1–9.

[54] Guillaume Tauzin et al. "*giotto-tda*: a topological data analysis toolkit for machine learning and data exploration". In: *J. Mach. Learn. Res.* 22 (2021), Paper No. 39, 6. ISSN: 1532-4435.

[55] Wikipedia contributors. *Black box — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Black_box&oldid=1053753402. [Online; accessed 20-December-2021]. 2021.

[56] Wikipedia contributors. *Cross entropy — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Cross_entropy&oldid=1045652153. [Online; accessed 18-December-2021]. 2021.

[57] Wikipedia contributors. *Curse of dimensionality — Wikipedia, The Free Encyclopedia*. [Online; accessed 18-December-2021]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Curse_of_dimensionality&oldid=1059361687.

[58] Wikipedia contributors. *Data dredging — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Data_dredging&oldid=1058121117. [Online; accessed 15-December-2021]. 2021.

[59] Wikipedia contributors. *Logistic regression — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Logistic_regression&oldid=1060778010. [Online; accessed 22-December-2021]. 2021.

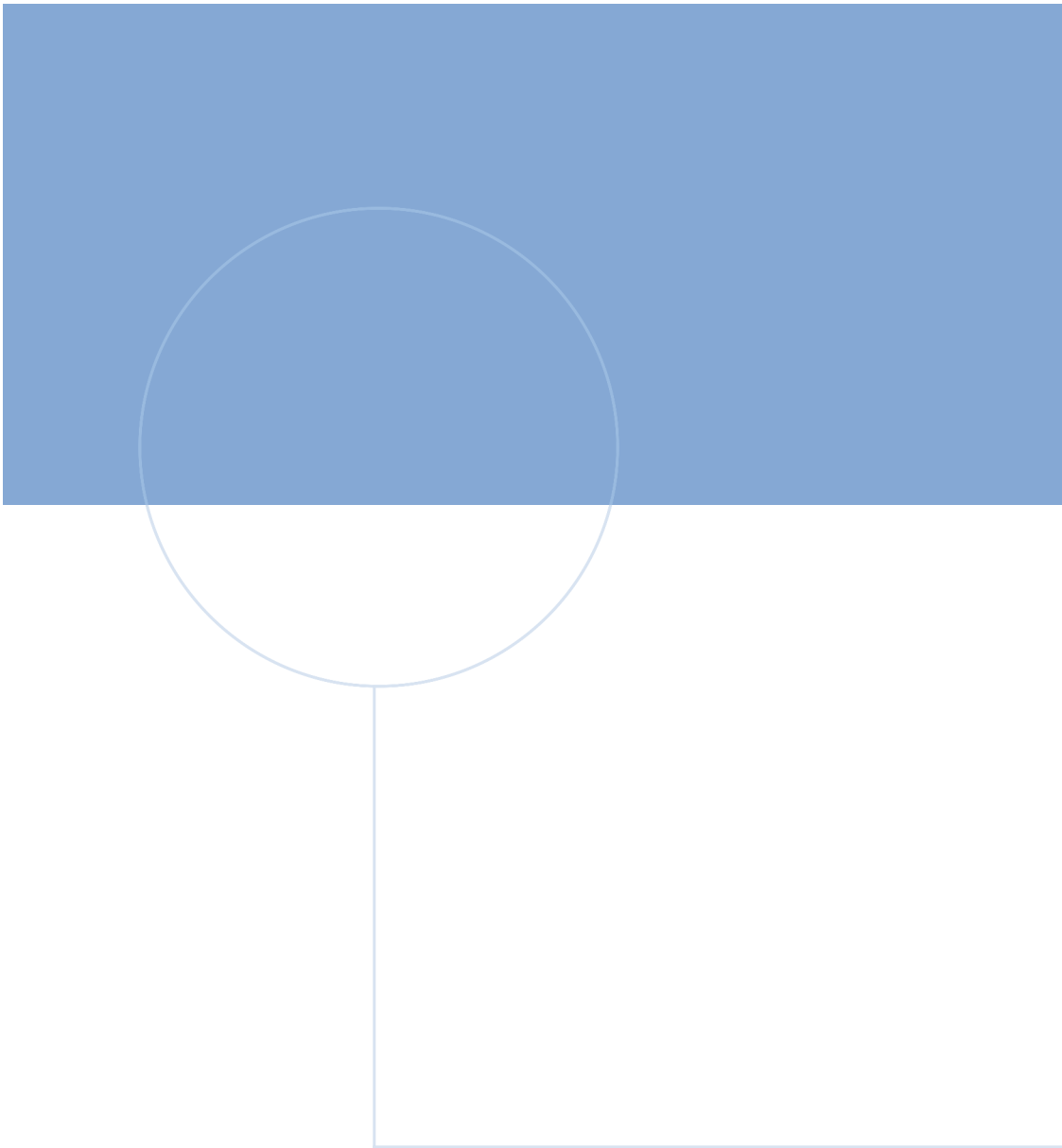
[60] Wikipedia contributors. *Occam's razor — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Occam%27s_razor&oldid=1060371999. [Online; accessed 20-December-2021]. 2021.

[61] Wikipedia contributors. *One-hot — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=One-hot&oldid=1059604951. [Online; accessed 23-December-2021]. 2021.

[62] Wikipedia contributors. *Simplex — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Simplex&oldid=1060160718. [Online; accessed 22-December-2021]. 2021.

[63] Chiyuan Zhang et al. "Understanding Deep Learning (Still) Requires Rethinking Generalization". In: 64.3 (Feb. 2021), pp. 107–115. ISSN: 0001-0782. DOI: 10.1145/3446776. URL: https://doi.org/10.1145/3446776.

[64] Afra Zomorodian and Gunnar Carlsson. "Computing persistent homology". In: *Discrete Comput. Geom.* 33.2 (2005), pp. 249–274. ISSN: 0179-5376. DOI: 10.1007/s00454-004-1146-y. URL: https://doi.org/10.1007/s00454-004-1146-y.



NTNU

Norwegian University of
Science and Technology