École polytechnique fédérale de Lausanne

Semester project spring 2025

Master in Computational Science and Engineering

Using Algebraic Topology and Graph Neural Networks in Multi Embedding Fusion Models for classifying neuronal morphologies

Author:
Jan Zgraggen

Supervisor:
Lida Kanari

Contents

List of Figures

List of Tables

Abstract

This project explores a unified framework for neuronal morphology classification making use of Topological Data Analysis (TDA). Building on the foundations of algebraic topology, more precisely persistent homology and the Topological Morphology Descriptor (TMD) algorithm, the project investigates how diverse representations — ranging from topological summaries to graph-based and morphometric embeddings - are able to distinguish between cell types and capture complementary structural features of neuronal data.

By reproducing and extending results from recent literature, the project confirms the effectiveness of TMD-based descriptors and demonstrates the added value of deep learning approaches, particularly Graph Convolutional Networks. An integral part of this work is the development of a Multi-Embedding Fusion model, which integrates multiple representation modalities through pre-classification layer concatenation to achieve improved classification accuracy. Applied to the Janelia L5 Pyramidal Cells dataset, this approach analyzes the complementarity of different cell representations and raises the benchmark accuracy from 81% to 94%, highlighting the potential of embedding fusion for this task.

The results highlight the usefulness of TMD-based representations as well as the importance of multimodal feature integration for understanding complex biological structures. They also offer a robust, extensible pipeline for future work in neural morphology classification.

1 Introduction

Understanding the structure-function relationship of neurons is fundamental to neuroscience. Neuronal morphology, particularly the complex branching patterns of neuronal cells, plays a key role in defining cellular function and connectivity. However, consistent classification of these morphologies remains challenging due to subjectivity in expert labeling and the inherent complexity of neuronal shapes.[7]

Recent advances leverage mathematical frameworks such as algebraic topology—more precisely, persistent homology—and various machine learning methods to objectively quantify and classify neuron morphologies. Notably, topological tools like the Topological Morphology Descriptor (TMD) have demonstrated that the TMD patterns of apical dendrites alone can robustly distinguish neuronal types.[4, 6]

In this project, I reproduce and explore methods presented in two recent studies by Kanari et al.[7, 6], which apply topological and statistical tools to neuronal morphology. The goal is to validate and classify neuron types using a combination of persistent homology, graph-based, and traditional machine learning classifiers. Furthermore, ideas are elaborated on how these methods, originating from different modalities, can be integrated to achieve better performance on selected classification tasks. By doing so, this project contributes to the development of objective, scalable, and interpretable frameworks for neuronal classification based on morphology.

2 Background

2.1 Building Blocks of Algebraic Topology

The goal of this section is not to give a complete description of topics and concepts of algebra as found in [8], or topology and algebraic topology as found in [1] and [9], but rather to introduce important notation and develop an intuition for someone having the same prerequisites as I had at the start of this project[1]. It lays the foundation for the core concepts of this work, such as Persistent Homology and its application to the Topological Morphology Descriptor, while keeping the theory as self-contained as possible.

2.1.1) Monoids and Groups

In order to understand concepts of Persistent Homology and its building blocks, such as topological spaces and their functions and maps (e.g. homomorphisms), I shall introduce the algebraic concept of groups in a broader context, starting with the monoid as a superclass of a group.

A monoid is defined as follows:

Definition 1. A monoid is a triple (M, p, I) in which M is a non-vacuous set, p is an associative binary composition (or product) in M, and I is an element of M such that $p(I, a) = a = p(a, I)\ \forall a \in M$ [8].

A trivial example is the monoid $(\mathbb{N}, +, 0)$ [8]. A group G is a special case of a monoid.

Definition 2. A *group* G (or (G, p, I)) is a monoid (Def. 1) all of whose elements are invertible. [1]

Again, a trivial example is the group $(\mathbb{Z}, +, 0)$ [8]. However, the concept is not restricted to number sets. Without rigorously introducing the required formalism of a *homotopy group*[2] or a *fundamental group*[3], we shall examine an example given in [1]:

Example 1. Imagine two linked circles, A and B, in $\mathbb{R}^3$: Let A be fixed, and let B_i, a "species" or "type" of B, be part of a group: $G = (B, p, B_0)$.

We define the elements of B such that the subscript indicates the number of windings around A, and the sign denotes the orientation, as shown in Figure 1. To enable algebraic operations, we assume that all B_i

[1] Engineering graduate student without a special background in mathematics
[2] A form of group defined in a topological space
[3] The first-order homotopy group

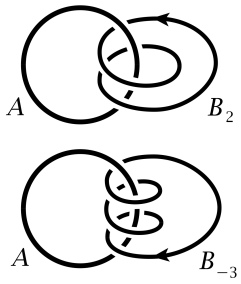


Figure 1: Examples of elements in the set B_i of the group of linked circles. [1]

originate from the same point x_0 in $\mathbb{R}^3$. Thus, the operation p (here conveniently interpreted as addition: $+$) in the group determines how two elements of B_i are connected at the point x_0. The result of the operation depends on the orientation of the elements involved. The algebraic structure behaves as expected. The intuition is illustrated in Figure 2.

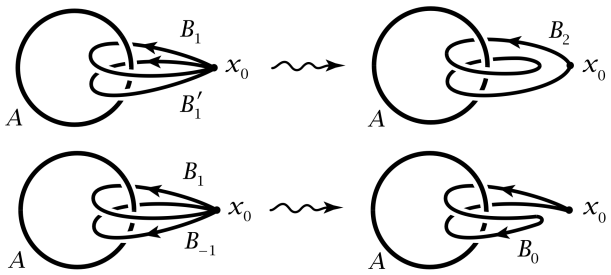


Figure 2: Visualization of the addition operation p on elements of B. Shown are $p(B_1, B_1) = B_2$ and $p(B_1, B_{-1}) = B_0$ [1].

The resulting algebra, under additive notation ($p := +$) and with B_0 as the identity element I, gives us $B_n + B_m = B_{m+n}$ [1].

Note that depending on the structure of the set G, a multiplicative notation for $p(\cdot, \cdot)$ may be preferred.

2.1.2) Properties and Maps of Groups

For the following concepts, we will use multiplicative notation. We briefly introduce relevant properties and maps of groups, which recur throughout the development of concepts leading to Persistent Homology. First, we introduce the notion of commutativity:

Definition 3. A group G or (G, p, I) (Def. 2) is called an *abelian* group if $\forall x, y \in G$ one has $p(x, y) = p(y, x)$ [8].

Additionally, one can define a property characterized by the existence of a subset that acts as a basis. That is, every element of the group can be uniquely expressed as a combination of these basis elements under the group operation $p(\cdot, \cdot)$.

Definition 4. A group G (Def. 2) is called *free* if there exists a set $S \subset G$ such that every element $a \in G$ can be written uniquely as $a = \prod_i s_i^{e_i}$, with $s_i \neq s_j$ for $i \neq j$, $s_i \in S$ $\forall i$, and $e_i \in \mathbb{Z} \setminus \{0\}$ $\forall i$ [8].

A formal method to construct such a basis is found in [8].

Definition 5. For a group G,the elements of S form Def. 4 are called *generators*.

Definition 6. A group G is called *free abelian* if G is free (Def. 4) as well as abelian (Def. 3).

Having developed an intuition about groups, we now discuss maps between groups, $G \to G'$. More precisely, we are interested in maps that preserve the group operation p. Such maps are called homomorphisms and are formally defined as follows:

Definition 7. A *homomorphism* $f : G \to G'$ is a map such that $f(x \cdot y) = f(x) \cdot f(y)\ \forall x, y \in G$ [9].

Such a homomorphism automatically satisfies the relations $f(I) = I'$ and $f(x^{-1}) = f(x)^{-1}$, where I and I' are the identity elements in G and G', respectively, and the exponent -1 denotes the inverse [9].

A homomorphism has two associated sets of interest: the kernel and the image.

Definition 8. The *kernel* of a homomorphism f is the set of elements that are mapped to the identity element I' of G':

$$\mathrm{Ker}(f) = \{a \in G \mid f(a) = I'\}[9].$$

Definition 9. The *image* of a homomorphism f is the set of all elements in G' that have a pre-image in G:

$$\mathrm{Im}(f) = \{f(a) \mid a \in G\}[9].$$

In the following, we mainly work with boundary homomorphisms—that is, maps that send their arguments to their boundaries.

Definition 10. A homomorphism $f : G \to G'$ (Def. 7) is called a *boundary homomorphism* if $G' = \partial G$, the boundary of G. It is commonly denoted as ∂ or ∂_p for $\dim(G) = p$. A fundamental property is:

$$\partial_p \circ \partial_{p+1} = 0[2].$$

2.1.3) Homology

The broad idea of Homology is to detect cycles and boundaries algebraically in order to classify topological spaces [1].

To introduce the concept of Homology, we first define the notion of complexes. For simplicity, we restrict ourselves to *CW-complexes*, from which we construct *chain complexes*, central to (persistent) Homology.

2.1.4) Complexes

A cell complex, or *CW-complex*[4], is a space composed of cells of different dimensions—points (0-cells), edges (1-cells), faces (2-cells), etc.—attached to each other. Formally:

Definition 11. A n-dimensional *CW-complex* is a topological space X_n constructed inductively from n-cells C_n of various dimensions. Each n-cell C_n is defined via the inductive construction of the CW-complex. The process begins with a set of 0-cells C_0, i.e., zero-dimensional points. Iteratively, $(n+1)$-cells C_{n+1}—defined as continuous maps with n-cell boundaries—are attached to C_n to form higher-dimensional faces of the CW-complex [1, 10].

A trivial example is an "empty triangle"[5], which consists of four 0-cells (corners) and three 1-cells (edges). An empty circle can also be viewed as a CW-complex, by taking any point as the 0-cell and the rest of the circle as a 1-cell—represented as a continuous map from its boundary to itself.

[4] CW comes from C: Cell and W: Weak Topology

[5] This is a special case of CW-complex. More generally, one defines special cases such as *simplicial complexes* when the cells are simplicial (line segments, triangles, polyhedra, etc.). Equivalently, cubical or other complexes can be defined [10], [1].

2.1.5) Chain Complexes

To build up the concepts of chain complexes and homology groups, we consider an example from HATCHER 2002 [1], which also sheds light on the intuition behind group-theoretic concepts.

We first define the notions of *p-chain* and *chain group*. Their intuition is illustrated in Example 2.

Definition 12. A *p-chain* is a p-dimensional object constructed as a formal sum of p-cells [1, 2].

Definition 13. A *chain group* is a free abelian (Def. 6) group whose elements are p-chains σ. That is [1, 2]:

$$C_p(X) = \bigoplus_{\sigma \in X_p} \mathbb{Z}\langle\sigma\rangle,$$

where $\mathbb{Z}\langle\sigma\rangle$ indicates that the p-chains have integer coefficients.

Example 2. Consider the graph X_1 in Figure 3. Let's examine the chain group defined by the loops on the graph, such as (in additive notation)[6] $a - b$, which represents a loop through a in the positive direction, returning through b in the negative direction.

We "abelianize"[7] the group by considering $a - b$ and $-b + a$ as equivalent.

As discussed in Section 2.1, an example of a member of this group is any closed loop along the graph, such as $a - c + b - d + c - a$. To enable algebraic notation resembling linear combinations, we identify loops that form the same cycles. For instance, the cycle defined by $(a - c) + (b - d)$ is the same as $(a - d) + (b - c)$, so we can write an arbitrary cycle as $ka + lb + mc + nd$, with the cycle condition $k + l + m + n = 0$[8] [1].

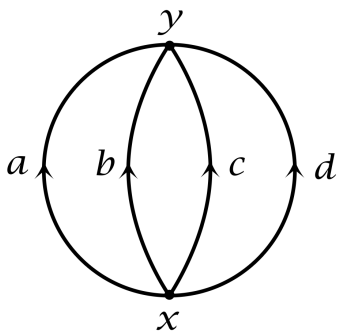


Figure 3: Graph of 1-CW-complex X_1 with points x, y and directed edges a, b, c, d [1].

We now relate the 1-cells (edges: a, b, c, d) and 0-cells (points: x, y) within the defined group. Let the chain group C_1 consist of cycles as above, and C_0 be the chain group with basis $\{x, y\}$, parametrized as $px + qy$, where $p + q = 0$ due to the cycle condition.

Define a homomorphism $\partial_1 : C_1 \to C_0$ that maps each basis element of C_1 to its boundary, e.g., $y - x$[9]. For any $g = ka + lb + mc + nd \in C_1$, we then have:

$$\partial_1 g = (k + l + m + n)x - (k + l + m + n)y.$$

For a general graph with $g = \bigoplus_{x \in \text{base}(C_1)} \mathbb{Z}\langle x\rangle$, the homomorphism becomes:

$$\partial_1 g = \bigoplus_{x \in \text{base}(C_0)} \left(\bigoplus_{\substack{a \in g \\ a \in \text{base}(C_1)}} \mathbb{Z}\langle a\rangle \right) \text{[1]}. \tag{1}$$

[6] For better intuition on linearity, we use additive notation throughout this example. The equivalent multiplicative notation for $a - b$ is ab^{-1}.

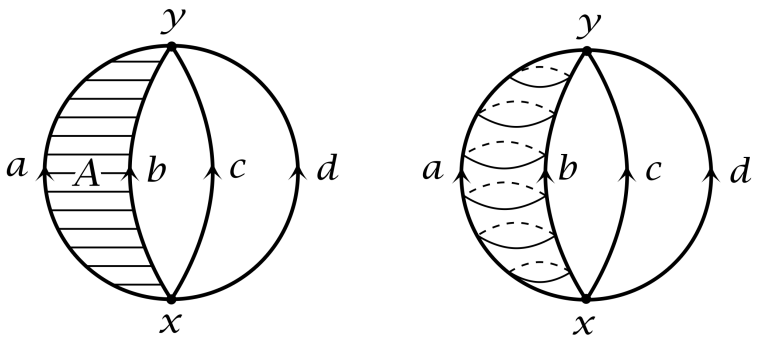
[7] We consider the abelian group for simplicity, i.e., we ignore the starting point.

[8] This arises from the condition that each node must be entered as often as it is exited.

[9] Choosing $x - y$ instead would merely reverse the sign.

The point of this lengthy example becomes clear when we step up in dimensions. Let us first add a 2-cell A between a and b, creating a two-dimensional space X_2, as seen in Figure 4a.

We define the *Chain Group* C_2 as the group with the single basis element A, having boundary $a - b$ [1].



(a) Visualization of a 2-CW-Complex with surface A (X_2).

(b) Visualization of a 3-CW-Complex X_3 with surfaces A, B.

Figure 4: Graphs for two- and three-dimensional complexes [1].

Similarly, one can construct a space X_3 with *Chain Group* C_2 spanned by A and B, or a space X_4 by adding a 3-cell between A and B, which again forms a *Chain Group* C_3 with one basis element, as seen in Figure 4b.

We can now define the homomorphisms $\partial_2 : C_2 \to C_1$ and $\partial_3 : C_3 \to C_2$, which, similar to ∂_1, map each respective chain group to its corresponding boundary—$a - b$ and $A - B$, respectively. Generalizing Equation 1 to p dimensions, the homomorphism ∂_p applied to $g \in C_p$ is given by:

$$\partial_p g = \bigoplus_{x \in \text{base}(C_{p-1})} \left(\bigoplus_{\substack{a \in g \\ a \in \text{base}(C_p)}} \mathbb{Z}\langle a \rangle \right) \tag{2}$$

We now have the required concepts to introduce a Chain Complex:

Definition 14. A *Chain Complex* (C_*, ∂_*) of a p-dimensional space X is defined as a sequence of chain groups and their boundary maps:

$$(C_*, \partial_*) = C_p \xrightarrow{\partial_p} C_{p-1} \xrightarrow{\partial_{p-1}} \dots \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0 \tag{3}$$

Hence, for the spaces defined in Example 2, we have, e.g., the chain complex of X_2:

$$C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0. \tag{4}$$

2.1.6) Homology Groups

Let us analyze the previously defined homomorphisms with respect to the chain complex in Equation 4 and Figure 4a.

The kernel of the map ∂_1, denoted $\ker(\partial_1)$, consists of all $g \in C_1$ that map to the identity element in C_0, i.e., zero under addition. Hence, all cycles (closed loops) are elements of $\ker(\partial_1)$. It is evident that the cycles $a - b$, $b - c$, and $c - d$ form a basis for $\ker(\partial_1)$.

The image of the map ∂_2, denoted $\text{Im}(\partial_2)$, is the boundary of A, i.e., $a - b$ [1].

We now consider the quotient group[10] $\ker(\partial_1)/\,\text{Im}(\partial_2)$ and observe that this group has two generators.

[10] For a rigorous definition, Hatcher [1] refers to [11] and [12]. Intuitively, think of the quotient a/b ("a modulo b") where $\text{base}(b) \subset \text{base}(a)$ as the group formed by the basis $\text{base}(a) \setminus \text{base}(b)$.

The intuition is that the number of generators of this group corresponds to the number of 1-dimensional holes. $\ker(\partial_1)$ includes all cycles (holes), and $\text{Im}(\partial_2)$ represents those cycles that are the boundaries of higher-dimensional cells (which should not be counted as "holes"). In this example, $\ker(\partial_1)$ includes three holes, and one (the $a-b$ loop) is the boundary of a 2-cell A, so it is subtracted out.

Another way to think of this is to consider p-dimensional cycles Z_p (which form holes) and subtract those that are boundaries of $(p+1)$-cells B_p. This leads to the formal definition of the p-th Homology group [1]:

Definition 15. The p-th *homology group* of a space X, denoted $H_p(X)$, is defined as [1]:

$$H_p(X) = \ker(\partial_p)/\,\text{Im}(\partial_{p+1}) \tag{5}$$

The number of p-dimensional holes is then formalized by the Betti number [1, 2]:

Definition 16. The *Betti number* $\beta_p(X)$ is defined as [2]:

$$\beta_p(X) = \text{rank}(H_p(X)) \tag{6}$$

2.1.7) Persistent Homology

Persistent Homology is the study of how topological features *persist* throughout the "growth" of spaces.

Filtrations To understand what is meant by the "growth" of a space, we introduce filtrations.

Definition 17. A *filtered space* is a nested sequence of subspaces starting with the empty set and ending with the complete space [2]:

$$\varnothing = X_0 \subseteq X_1 \subseteq \ldots \subseteq X_m = X \tag{7}$$

Growth is then interpreted as the transition from X_i to X_{i+1}.

Persistence Module Applying the homology functor to a filtration yields a *persistence module*, a sequence of linear maps [2]:

$$0 = H_p(X_0) \to H_p(X_1) \to \ldots \to H_p(X_m) = H_p(X) \tag{8}$$

This persistence module forms the basis for analyzing persistent homology.

We can observe the "growing" space and detect the point in the filtration where a new generator[11] F appears in the homology group of the current subspace. This point is called the **birth**. The generator persists through subsequent spaces X_j for $j > i$ until the j-th map in the persistence module maps it to zero (trivial), called its **death**.

More generally, a persistence module can be decomposed into indecomposable summands (the generators) of the form:

$$0 \to F \xrightarrow{I} \ldots \xrightarrow{I} F \to 0 \tag{9}$$

where I denotes the identity homomorphism [2].

Consider the example of the persistence module of a space X_2, which is decomposed into two indecomposable components:

$$\begin{array}{cccccc} & 0 \to & 0 \to & F^a \to & F^a \\ \oplus & 0 \to & F^b \to & F^b \to & 0 \\ \hline = & 0 \to & H_p(X_1) \to & H_p(X_2) \to & H_p(X_3) \end{array} \tag{10}$$

We find two indecomposable summands: chains F^a and F^b. Here, $H_p(X_1)$ is generated by F^a, $H_p(X_2)$ by $\{F^a, F^b\}$, and $H_p(X_3)$ by F^b.

[11] Basis element of a group.

Examples and Applications The set of intervals (birth, death) serves as a descriptor of a topological space and is used in Topological Data Analysis (TDA).

In order to understand filtrations and the persistence module, we will now examine several illustrative examples.

Example 3. *Continuous Space*
We consider the continuous space X^{cont} shown in Figure 5.

Definition 18. We define the *filtration function* $f(\epsilon, \boldsymbol{\theta}) : X \to X_\epsilon \subseteq X$ with expansion parameter ϵ and parameters $\boldsymbol{\theta}$, to be a function that maps a space to subspaces such that any sequence $\{X_\epsilon\}$, with ϵ increasing, forms a filtered space.

The filtration function in this example is realized by growing the space in the z-direction (height), with:

$$X_i^{\text{cont}} := \{x = (x_1, x_2, x_3) \in \mathbb{R}^3 \mid x \in X^{\text{cont}} \text{ and } x_3 \in [a, h_i]\} \tag{11}$$

where $h_0 = a$ and $h_{i+1} > h_i$.

Although we developed the formalism only for cell complexes, the analogy is still visible here. At the critical points $a, b, \ldots, f$, generators appear or disappear (birth and death). This is visualized by the homotopy equivalences of the filtration, which change in different height regimes.

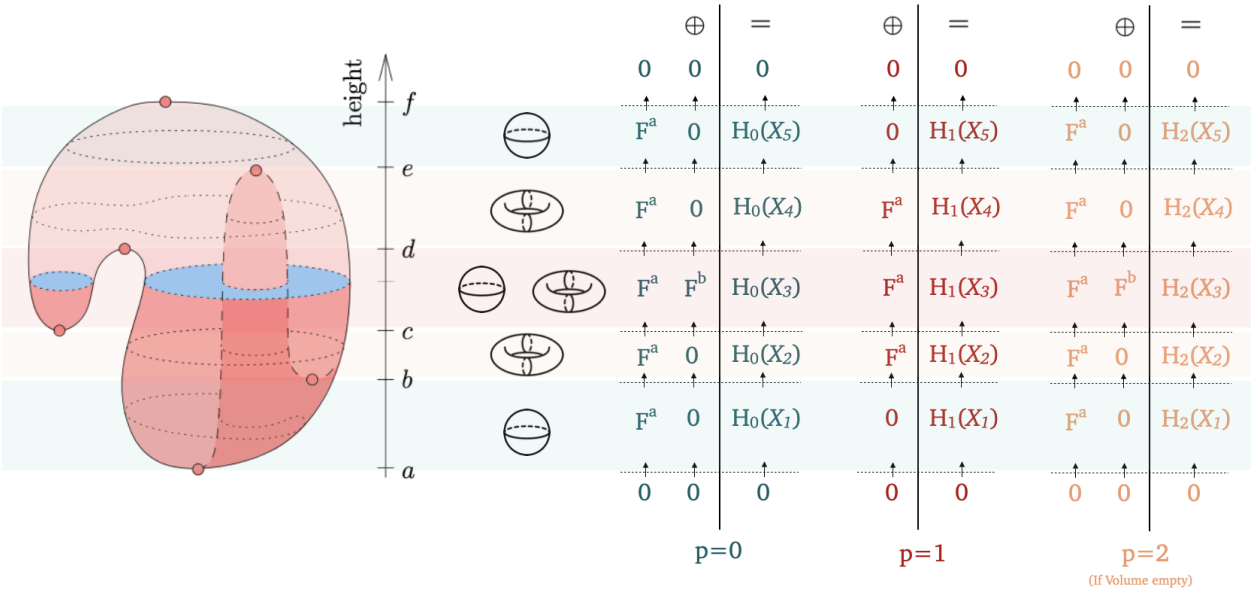


Figure 5: Visualization of Persistent Homology of a Continuous Space with Homotopy Equivalent Spaces and the indecomposable sums of the Persistence Module [2].

Note: The filtration function need not be as simple as height; it only needs to fulfill the definition of a filtration. The way the space "grows" can be chosen arbitrarily.

Example 4. *Point Cloud Data*
We can construct a filtered space of CW complexes from point clouds using the Čech or Vietoris–Rips complex construction.

Definition 19. Given a collection of points $\{x_\alpha\}$ in Euclidean space $\mathbb{E}^n$, the Čech complex $\mathcal{C}_\epsilon$ is the abstract simplicial complex whose k-simplices are determined by unordered $(k+1)$-tuples of points $\{x_\alpha\}_0^k$ whose closed $\epsilon/2$-ball neighborhoods have a point of common intersection [3].

A *Vietoris–Rips* complex $\mathcal{R}_\epsilon$ has a slightly looser constraint: pairwise distances within ϵ suffice. This version is often easier to compute in practice [3].

As ϵ grows, higher-order cells are formed (birth) when the ϵ-balls begin to overlap ($\mathcal{C}_\epsilon$), or when pairwise distances fall below ϵ ($\mathcal{R}_\epsilon$). Cells are destroyed (death) when integrated into a higher-order cell—e.g., a third point comes into range for two connected points, forming a triangle.

A visualization of a sequence of Rips complexes forming a filtration is shown in Figure 6.

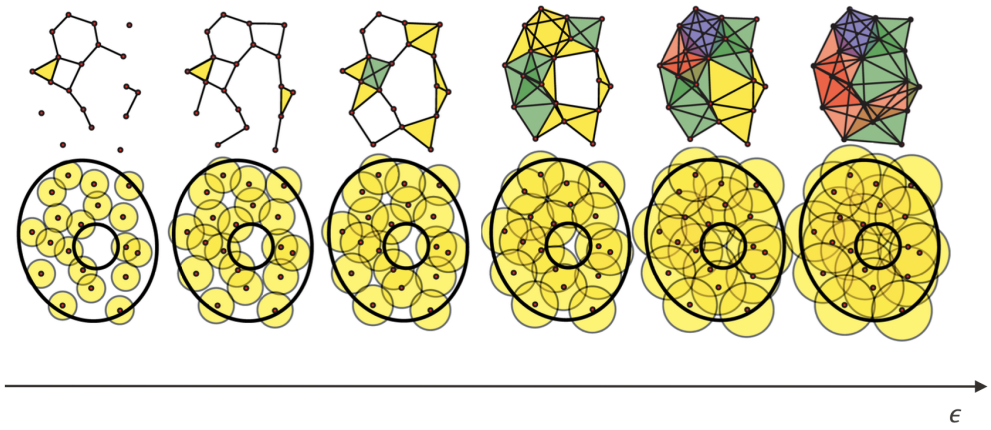


Figure 6: Visualization of Persistent Homology of Vietoris–Rips Complex $\mathcal{R}_\epsilon$ [3].

Example 5. *Neural Cells*
An applied example is that of neural cells, which can be represented as rooted trees embedded in $\mathbb{R}^3$. We create a filtration by "growing" the space starting from an empty sphere of radius $r - \epsilon$. As ϵ increases, the imaginary sphere around the root expands outward (Figure 7).

Similar to Example 3, we observe the topological changes of the homotopy-equivalent space at critical points. At these points (e.g., branch tips or junctions), a branch is "born" when it enters the volume, and "dies" upon merging with another branch[12].

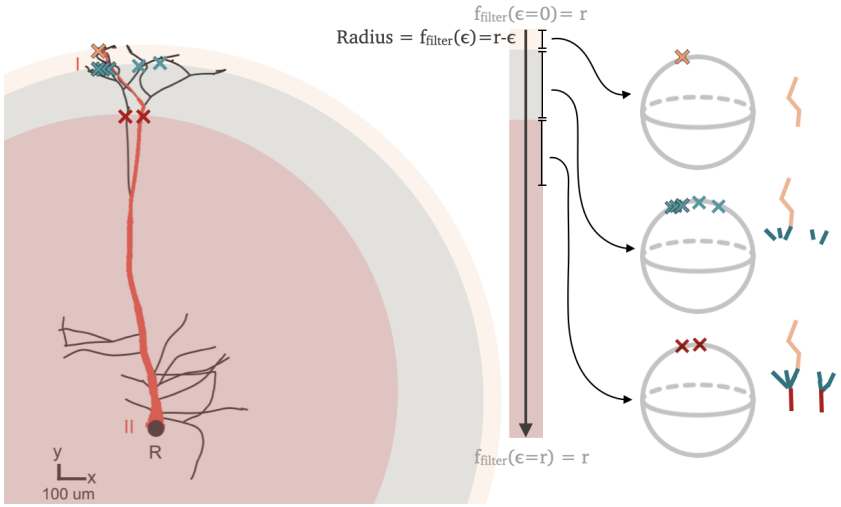


Figure 7: Visualization of Persistent Homology of Neural Tree [4]

[12]This procedure is clarified with the TDM algorithm, cf. subsubsection 2.2.3)

2.2 Topological Data Analysis (TDA)

2.2.1) Data

An intuitive way of representing multidimensioal data is to embed them in Cordinate spaces. However basic coodrinate spaces of data are not natural, and thus qualitative information is needed make use of the topology of data.[13]. In [14], the theoretical foundation for various methods to use of the topology of point cloud data is discussed and established. For practical reason we have been restricting ourselves to discuss persistent methods. In the following subsections we show how to discretize such methods to numerical representations[13] and later to vectors, using point cloud data or valorizing the inductive bias of the structure of the data (e.g Tree shape of Neuronal Cells).

2.2.2) Barcodes and Diagrams

Barcodes and Persistence Diagrams

Definition 20. *Persistance Barcodes* or simply *Barcodes* are defined as a set of tuples (a, b) with $a, b \in \mathbb{R}$ such that for all tupples of the Barcode, a represents the value of the expantion parameter ϵ of the filtration function used to produce a filtered space [3, 4].

The name Barcode stems form the fact that this resembles a set of real valued closed intervals, which if ordered (e.g. by lenght) can be visualized as a Barcode [3, 4].

The intervall can also be represented as 2D points:

Definition 21. The *Persistence Diagram*, short PD is defined as the 2d-plot,where each tuple (a, b) in the set of the barcode represents a point in $\mathbb{R}^2$ with coordinates $x = a, y = b$.

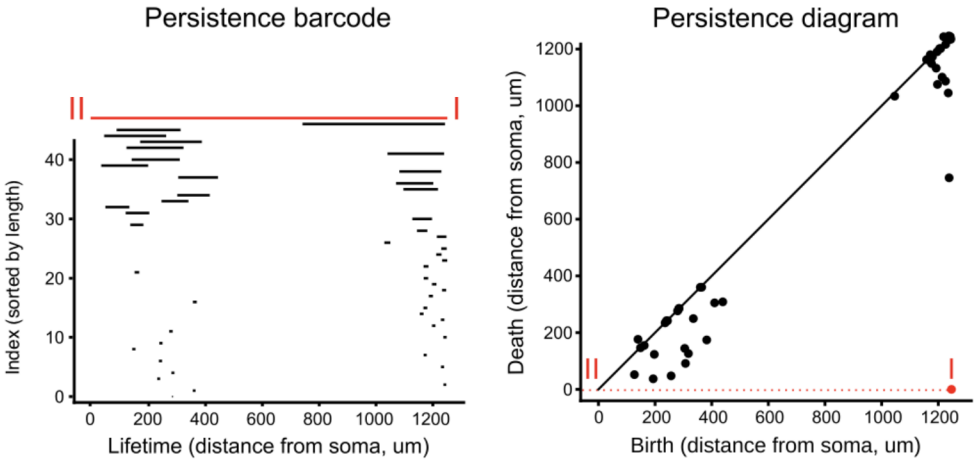


Figure 8: Barcode and Persistence diagram of the Neuron cell form Example 5 [4].

2.2.3) Barcode generation - Topolicical Morphology Descriptor (TDM)

Obtaining a barcode from Point Cloud data[14] with no intrinsic natural way of grouping the points[13] can be achieved for example via sequences of Vietori-Rips Complexes $\mathcal{R}_\epsilon$, or sequenzes of Čech complexes $\mathcal{C}_\epsilon$ as introduced in Example 4 or similar methods[13, 14].

If one has tree-structured data rather than complexes, Topological Morphology Descriptor (TMD) is an algorithm that generates a set of barcodes. The algorithm is described in detail by Kanari et. al.[4].

[13] Barcodes, PD's see subsubsection 2.2.2)

If there exists some other inductive biases or structures of the data other forms of barcode generation may be proposed.

Vectorizations of persistance diagrams *Persistance Image*
Intuitively the persistence image is a way of extrapolating the Persistence Diagram Plot to a 2D image. This is done by transforming the PD to a scalar function: $\mathbb{R}^2 \to \mathbb{R}$, which is then discretized. The function is achieved by summing over the product of a weighting and a distribution function of the points in the diagram [5]. Formally:

Definition 22. For a persistence diagram f, weighting function $w \in C^0(\mathbb{R}^{\not\vdash})$ with $\mathbb{R}^2 \to \mathbb{R}, w(x,0) = 0, w$: piecewise differeciable and probability distribution $\phi_u : \mathbb{R}^2 \to \mathbb{R}$ with mean $u = (u_x, u_y)$ the corresponding *persistence surface* ρ_{dgm}: $\mathbb{R}^2 \to \mathbb{R}$ is the function [5][14]:

$$\rho_f(z) = \sum_{u \in f} w(u)\phi_u(z) \tag{12}$$

The Persistence image is given by the discretization of the persistence surface, given by

Definition 23. For a persistance diagram f,its persistence image is defined via the pixel wise integral over the surface. For a pixel p with value $I(p)$[5][15]:

$$I(p, \rho_f) = \iint_p \rho_f \; dxdy \tag{13}$$

Note that in literature the most widely used distribution is a gaussian distribution given by:

$$\phi_u = g_u = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-u_x)^2+(y-u_y)^2}{2\sigma^2}} \tag{14}$$

Figure 9 visualizes the steps involved in the creation of a persistence image.[16]

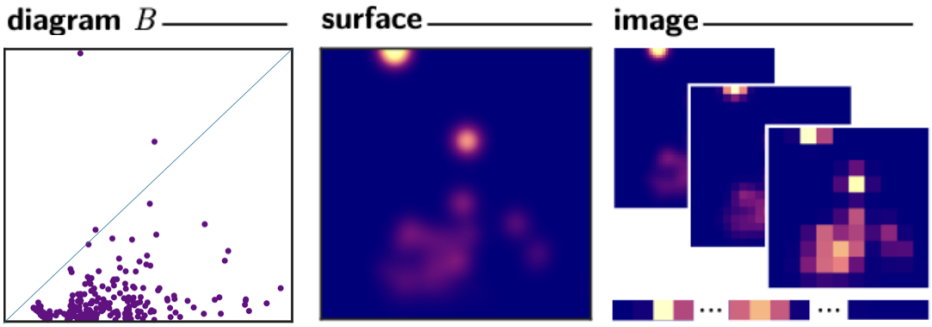


Figure 9: Schema for the persistence image creation. (Visualization by Adams et al.) [5].

[14] Adams et al. proposes to first apply a linear function $T(x,y) = (x, y-x)$ on the barcode tuples, here this transform is omitted for simplicity
[15] Stability results are discussed in detail in Adams et al.[5]
[16] As described in footnote 14 the linear transformation function is omitted for simplicity.

Persistance Landscape

A *persistence landscape* turns each birth–death pair (b, d) of a barcode into a triangle-shaped function that rises from b with slope 1 to the midpoint, then falls with slope -1 to d. At each point x, we take the k-th highest value among all these triangles and call it $\lambda_k(x)$. If there aren't k triangles active at x, then $\lambda_k(x) = 0$[15].

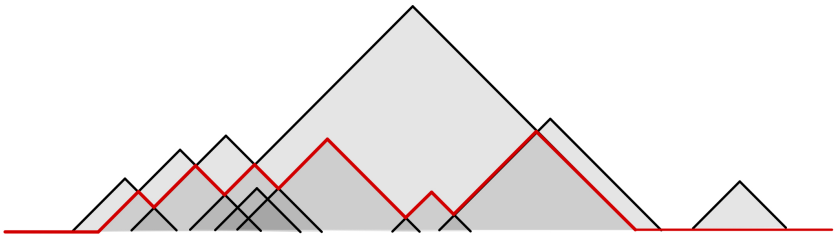


Figure 10: Schema for a persistence landscape with $k = 2$. The triangles represent the birth and deaths pairs of a persistence diagram.

Parwise distance Matrix

A plausible way of vectorizing a Persistence Diagram to a Vector which can be used for Machine Learning Techniques is to Compute the Pairwise distance matrix, and assign to each Diagram the Corresponding row of the Symmetric Matrix. I.e. the vector assignd to diagram f is the vector of distances to all other diagrams in the dataset. Various functions have been proposed in literature to compute pairwise distances of Persistance Diagrams. [2, 6, 16]. We shall state the distances used in [6] as well as in this project[17]:

Definition 24. The q^{th} Wasserstein distance between 2 Persistence Diagrams f and g is defined as [2] [6]:

$$W_q(f, g) = \inf_{\gamma} \left(\sum_{u \in f} \|u - \gamma(u)\|_\infty^q \right)^{1/q} \tag{15}$$

Definition 25. The Bottleneck distance is defined as [6, 2]:

$$W_\infty = \lim_{q \to \infty} W_q \tag{16}$$

These distances are computationally intensive, this is why a sliced version was proposed in literature:

Definition 26. Given $\theta \in \mathbb{R}^2$ with $\|\theta\|_2 = 1$, let $L(\theta)$ denote the line $\{\lambda\theta \mid \lambda \in \mathbb{R}\}$, and let $\pi_\theta : \mathbb{R}^2 \to L(\theta)$ be the orthogonal projection onto $L(\theta)$. Let f, g be two persistence diagrams (PDs), and define $\mu^\theta_{dgm} := \sum_{p \in dgm} \delta_{\pi_\theta(p)}$, $\mu^\theta_{dgm\Delta} := \sum_{p \in dmg} \delta_{\pi_\theta \circ \pi_\Delta(p)}$

where π_Δ is the orthogonal projection onto the diagonal. Then, the *Sliced Wasserstein distance* is defined as[6, 16] :

$$SW(f, g) := \frac{1}{2\pi} \int_{\mathbb{S}^1} W_1 \left(\mu^\theta_f + \mu^\theta_{g\Delta},\ \mu^\theta_g + \mu^\theta_{f\Delta} \right) d\theta \tag{17}$$

[17] Intuition as well as Stability analysis is found in the provided sources

2.3 Classification of Neuronal Morphologies

This section sets up the terminology and relevant processes in neuronal morphology classification.

2.3.1) Morphology Data

Data Origin Neuronal cell type clustering for gaining insight on connectivity and functionality has been studied in different brain areas.[6, 17] Throughout the brain, regions with different structures and organizations exist, containing different cell types. For example, the mammalian neocortex is structured in 6 different layers, L1 to L6, containing different types of pyramidal cells and inhibitory interneurons.[6, 18, 19, 20, 21]

Data Collection Cell morphologies of neuronal cells are obtained by manual or semi-automated[18] examination of electron microscopy.[20, 23] Data points containing position and thickness are extracted. The extracted data is stored by storing data points along with connectivity information, effectively allowing the production of a graph structure containing the reconstructed morphology.[6, 20, 23]

2.3.2) Classification

Classification of morphology types is typically done by visual inspection by experts in the field. Such classifications make it possible to distinguish different morphologies into groups of cell types. Nonetheless, stable groupings by statistical methods based on the morphology of the cells are required to robustify cell type classification, as visual inspection by individuals is subjective, prone to error, and can lead to ambiguous classification throughout experts. Various approaches have been proposed for achieving stable classification as well as automation of the latter. To support classification via ML frameworks, morphological data must be converted into compatible formats. Various feature extraction strategies have been proposed, typically falling into one of three broad categories: TMD (PI)-based descriptors[4, 6, 5], graph-based representations [7], and global morphology statistics, also referred to as morphometrics.[7, 5, 20, 24, 25, 26][19]

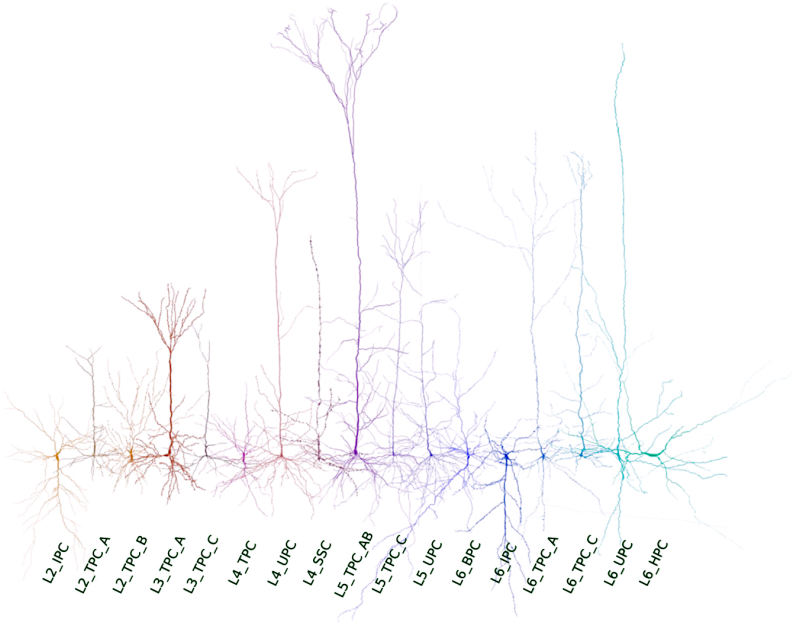


Figure 11: Visualization of different cell types form different layers (by Kanari et al.)[6]

[18] Recent literature has proposed machine learning methods for fully automated examination [22]
[19] In subsection 3.2, I describe the specific methods used to extract these features from neuronal reconstructions.

3 Methods

3.1 Datasets

The datasets used in this project are from the following two studies: Kanari et al. (2019) [6], which uses the dataset *Reconstructed* [27], and Kanari et al. (2024) [7], which uses the datasets *Pyramidal Cells* and *Interneurons* from the LNMC laboratory [20, 28], as well as *Pyramidal Cells* from the Janelia laboratory [23]. *Experiment 1* (paragraph 3.3.1)) uses the dataset *Reconstructed*. *Experiments 2* (paragraph 3.3.2)) and *3* (paragraph 3.4.3)) use the dataset *Pyramidal Cells* from the Janelia laboratory. The dataset *Pyramidal Cells* from the LNMC laboratory was used for testing.

These datasets consist of 3D reconstructed trees representing the axon, apical, and basal dendrites of neuronal cells from the rat somatosensory cortex. The reconstructions result in a set of connected points traced from image stacks of the 3D neuronal morphology, with each point defined by a 3D position (X, Y, Z) and diameter [7].

All reconstructions were performed by the respective dataset providers.

3.2 Feature Extraction

To enable classification of neuronal morphologies using machine learning, I extracted features using three complementary approaches: TMD-based descriptors, graph-based representations, and morphometrics. Each approach produces vector, tensor, or graph representations from raw morphological data, which are suitable for input into respective classifiers. The methods are detailed below.

3.2.1) MD-Based Vectorizations (Persistence Diagrams)

As described in subsubsection 2.2.3), one can obtain vector representations of morphology data by applying a vectorization to the Persistence Diagram obtained from the TMD algorithm.

I used the `TMD` [4] Python package to extract the Persistence Diagram via the TMD algorithm based on the radial distance function. I implemented the computation of the pairwise distance matrices using the `tda_toolbox` [29] Python package to calculate the pairwise distances (Wasserstein, Bottleneck, and Sliced Wasserstein). I extracted the Persistence Landscape using the `tda_toolbox` package and the Persistence Image using the `TMD` package.

3.2.2) Graph Representation

Morphology data is essentially already suited for a graph structure. However, depending on the exact structure of the morphology data, a processing step is usually still needed to bring the data into the expected format of a graph-based classifier such as a Graph Convolutional Neural Network (GNN) [30]. Furthermore, to make use of the positional encoding of the nodes given by the morphology, node, edge, and global features can be attributed to the graph structure to be utilized by a GNN architecture [30, 31]. Kanari et al. (2024) showed that radial distance was a more powerful node feature than coordinates [7].

I converted the morphology data into graph structures (nodes and edges) using the `morphoclass` [7] Python package. I used the `morphoclass` package to extract edge weights based on path length and node features based on the radial distance function.

3.2.3) Morphometrics

Another category of features consists of summary statistics that describe various aspects of cell morphology—such as size, path length, and branching patterns. This approach is commonly referred to as morphometrics. Several combinations of different morphometrics have been proposed in the literature [5, 20, 24, 25, 26, 32].

To ensure consistency with prior work, I used the collection of morphometrics (Appendix: Table 6 and Table 7) used by Kanari et al. [7].

I used the `morphoclass` package to extract the vectors containing the morphometrics of the respective morphologies.

3.3 Reproduction of State-of-the-Art Classification Results

3.3.1) Traditional ML with TMD-based Vectorizations

In a first experiment, results from the paper *Objective morphological classification of neocortical pyramidal cells* [6] are reproduced. Due to the numerous experiments presented in the paper, only results from the section *PCs in Layer 5* are presented.

Experiment 1.1 The first task of the experiment is to conduct a binary classification between UPC and TPC in cortical layer 5 (L5).

Experiment 1.2 The second task of the experiment is to perform multi-class classification among the L5 subclasses UPC, TPC_AB, and TPC_C.

Note: *The paper mentions that with resources only available after publishing, three TPC subclass types are distinguishable. As the data used for reproduction is updated and labeled with 3 TPC subclasses (A, B, and C), I merged A and B to reproduce the reported distinction between the subclasses mentioned in the experiment.*

I implemented a functional data loading pipeline that produces vectorizations based on TMD, as explained in subsubsection 3.2.1). I implemented an `SKTrainer`[20] class with class methods `train_crossvalidation` and `train_gridsearch`, using the `Sklearn` [33] Python package. This achieves the framework required for the reproduction of the results in Experiments 1 and 2.

For each vectorization method, I used the pipeline to optimize the classifiers: Decision Tree, SVC[21], and QDA[22] using grid search and cross-validation. Final results were averaged across all vectorization methods.

3.3.2) Graph Neural Network (GNN) based Classification

In the second experiment, results from the paper *Deep learning for classifying neuronal morphologies: combining topological data analysis and graph neural networks* [7] are reproduced. Again, I limit the experiments presented to results from the PCs in Layer 5 of the Janelia dataset [23].

Experiment 2 Experiment 2 aims to reproduce the Layer 5 (L5) cell types and subtypes classification (UPC, TPC_A, TPC_B, TPC_C) based on the Janelia dataset.

I used the CLI tool provided by the `morphoclass` [7] package to extract graph features as described in subsubsection 3.2.2). Next, using the CLI tool of `morphoclass`, I trained a graph neural network (GNN) model. Specifically, I used the `ManNet`[23] classifier provided by `morphoclass` [7], which consists of a graph-based embedder and a linear classification layer. This reproduces the exact classification workflow of Kanari et al. (2024) [7]. Furthermore, I conducted a sanity check by redoing the exact same experiment, but this time using the pipeline developed in subsection 3.4 instead of the CLI tool from `morphoclass` [7].

3.4 Multi-Embedding Fusion Model

3.4.1) Motivation

Kanari et al. (2024) [7] explored a variety of deep learning approaches for neural cell classification, using feature types such as Morphometrics, TMD (mainly Persistence Image), Graph representations, and DeepWalk embeddings. These were combined with machine learning models like PersLay, Decision Tree, XGBoost, CNN, and GNN. It was reported that GNN, TMD-CNN, and Morphometrics with a traditional classifier were the best-performing models in the study. Furthermore, Kanari et al. (2019) showed that, apart from

[20] SK stands for Scikit-learn [33], the Python package used for the implementation
[21] SVC: Support Vector Classifier
[22] QDA: Quadratic Discriminant Analysis
[23] Man: Multi Adjacency Network

Persistence Image, the vectorization methods based on the pairwise distance matrix—using Wasserstein, Bottleneck, and Sliced Wasserstein distances—as well as the Persistence Landscape, effectively distinguish classes of neuronal morphologies.

Hence, classification has proven successful using descriptors from all three origins (Graph, TMD, Morphometrics) discussed in subsubsection 2.3.2) and subsection 3.2. These representations from different origins may contain complementary information and thus motivate combining such representations. In this project, I explore the approach of a Multi-Modal-Embedding Fusion Model as a possible method for learning combined features from different origins.

Note: *While* `morphoclass` *[7] supports certain fused models (e.g., CNN + GNN or CNN + PersLay), the paper does not report results for them, likely due to lack of performance improvement*[24]*. Moreover, no experiments with multi-modal fusion models were conducted.*

3.4.2) MultiConcat Model

Using `torch` [34], `torch_geometric` [35], and inspiration from `morphoclass` [7], I implemented the `multiConcat` model along with a pipeline that combines feature extraction and model architecture, as illustrated in Figure 12.[25]

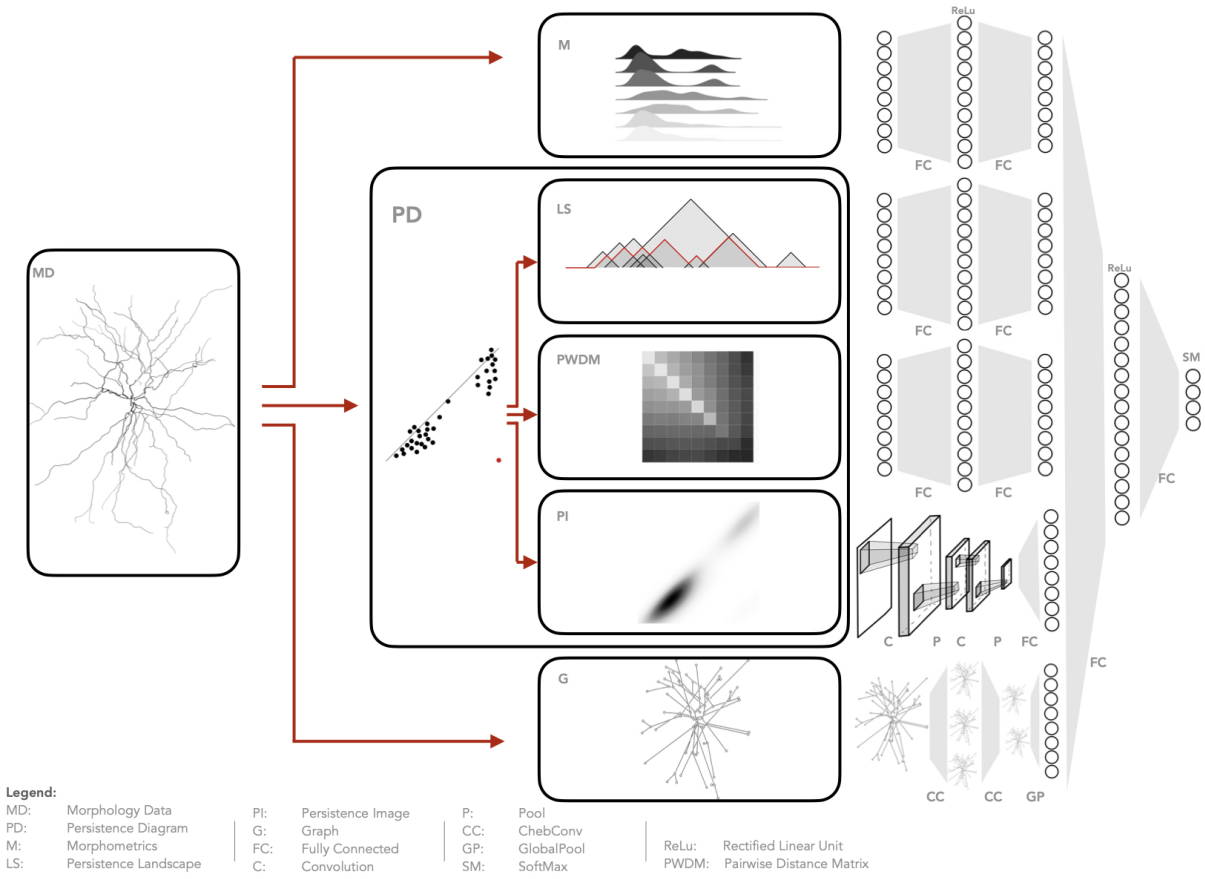


Figure 12: Schema of the Model Pipeline (Preprocessing and Architecture).

[24] Confirmed orally by the authors

[25] The graphic is created by the author. The images for MD, M, and PI are reused from [7].

Model Architecture The `multiConcat` model performs classification based on a pre-classification concatenation of uniform-dimensional embeddings. Each embedding is produced by a `ModalityEmbedder`, which processes a feature attribute according to its modality. Currently, three modalities are supported:

- `GraphEmbedder` is implemented as an adapted version of the `ManEmbedder` from `morphoclass` [7]. It uses two blocks of bidirectionally concatenated ChebConv layers, with a configurable hidden dimension in the second block, followed by a global pooling layer.
- `CNNEmbedder` is an extended version of the `CNNEmbedder` from `morphoclass` [7]. It consists of two 3×3 convolutional layers with 3 output channels, followed by a 2×2 max pooling layer. After flattening, a fully connected layer is appended to project the output to the target embedding dimension.
- `LinearEmbedder` embeds linear feature vectors into the embedding dimension using a small MLP with one hidden layer of configurable dimension.

The embedding stage is followed by concatenation of the embeddings, weighted by a learnable scalar parameter. The concatenated feature vector is passed to a classification head with one hidden layer and a softmax multiclass prediction on the logits.

Model Preprocessing I implemented a class `MorphologyDatasetManager` that extends the `MorphologyDataset` class from `morphoclass` [7] to support feature attributes of different modalities. I use functionality inspired by `morphoclass` [7] to extract the graph and morphometric feature attributes. I used the functionality described in subsection 3.2 to extract TMD-based feature attributes.

3.4.3) Experiments

Experiment 3.1 In the first task, I assess the classification performance of the `MultiConcat` model on the subclasses of L5 pyramidal cells from the Janelia dataset [23], using all available embeddings. Different embedding dimensions were evaluated. The results are compared to the benchmark set by the GNN `ManNet`, which achieved the best accuracy on layer 5 in the study by Kanari et al. [7].

Experiment 3.2 In a second assessment, `MultiConcat` is tested using different subsets of embeddings. The results are again compared to the benchmark set by `ManNet` from Kanari et al. [7].

Experiment 3.3 In the final experiment, the model is extended to output the learned scalar weights of each embedding type. This setup allows investigation of embedding preferences and evaluation of the optimal contribution of each embedding type and modality.

Note: *Due to its flexibility, I use the training loop implemented by the `morphoclass` CLI to train the `MultiConcat` model.*

3.5 Evaluation Metric

The evaluation procedure follows the method described by Kanari et al. [7].[26] A 10-fold cross-validation assesses the out-of-sample performance of the models. Stratification is used to ensure label distribution is consistent across splits. The evaluation metric is computed on the out-of-sample predictions produced by the trained models on each of the 10 validation splits. In order to ensure comparability with the literature, the *accuracy* score is chosen as the evaluation metric.[27]

The accuracy is defined as:

$$accuracy = \frac{1}{N}\sum_{i=1}^{N}\mathcal{I}_{\hat{y}_i=y_i} \tag{18}$$

where N is the number of samples, and $\mathcal{I}_{\hat{y}_i=y_i}$ represents the indicator function equal to 1 if the internal expression is true, and 0 otherwise.

[26] Due to the usage of the `morphoclass` CLI for the training loop, the evaluation metrics are adopted.
[27] The additional metrics computed by the `morphoclass` CLI training loop revealed no significant differences [7].

4 Results

4.1 Traditional ML with TMD-based Vectorizations

4.1.1) *Experiment 1.1*: Class Distinction

Among the tested classifiers, SVC yielded the best results in *Experiment 1*. Kanari et al. [6] report a 90% accurate class distinction on the 5th layer PC classes. Using cross-validation and a grid search over an SVM classifier, I nearly reproduce this result, averaging across the different vectorization methods used, as shown in Table 1.

4.1.2) *Experiment 1.2*: Detection of Subclasses

As presented in the paper, 3 subclasses are reported. I am able to confirm 3 stable subclasses using traditional ML models, achieving an accuracy of approximately 80%, as shown in Table 2.

Table 1: Cross-validated Accuracy of L5 Class Distinction TPC, UPC Using Different Distance Metrics

Distance Metric	Best Accuracy
Persistence Image	0.9435
Wasserstein	0.8876
Bottleneck	0.8690
Sliced Wasserstein	0.8871
Landscape	0.8749
Average Accuracy	**0.8924**

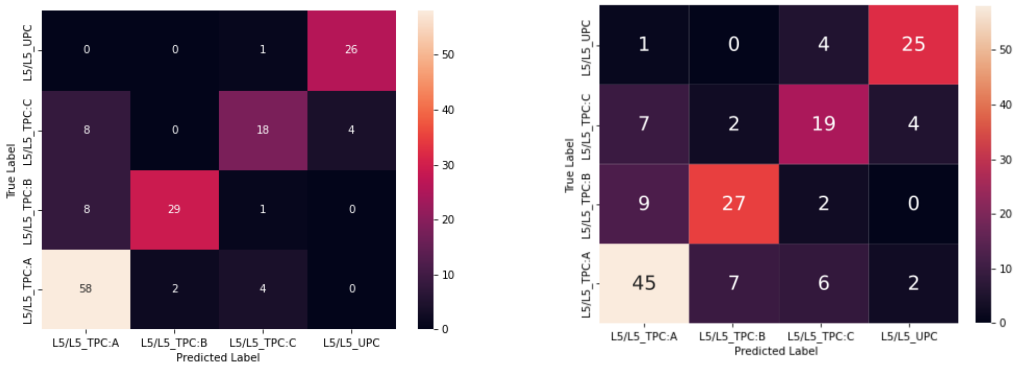
Table 2: Cross-validated Accuracy of L5 Class and Subclass TPC_{AB}, TPC_C, UPC Distinction Using Different Distance Metrics

Distance Metric	Best Accuracy
Persistence Image	0.8123
Wasserstein	0.8249
Bottleneck	0.7814
Sliced Wasserstein	0.8188
Landscape	0.7877
Average Accuracy	**0.8050**

4.2 Graph Neural Network (GNN) Based Classification

4.2.1) *Experiment 2*: GNN on L5 Pyramidal Cells (PCs)

Using the ManNet proposed by [7], I was able to reproduce the results on L5 Pyramidal Cells, achieving an accuracy of 82 ± 7% (versus 81 ± 7% reported in the paper).



(a) Confusion matrix for the reproduction of L5 PC classification with ManNet [7].

(b) Confusion matrix reported by Kanari et al. in [7].

Figure 13: *Experiment 2*

4.3 Multi-Embedding Fusion Model

No parameter optimization was performed initially. After observing rapid convergence, the number of epochs was reduced from 150 to approximately 30, and the learning rate was lowered from 0.005 to 0.001. This resulted in overall better performance and faster convergence across all experiments.

4.3.1) *Experiment 3.1*: Embedding Dimensions

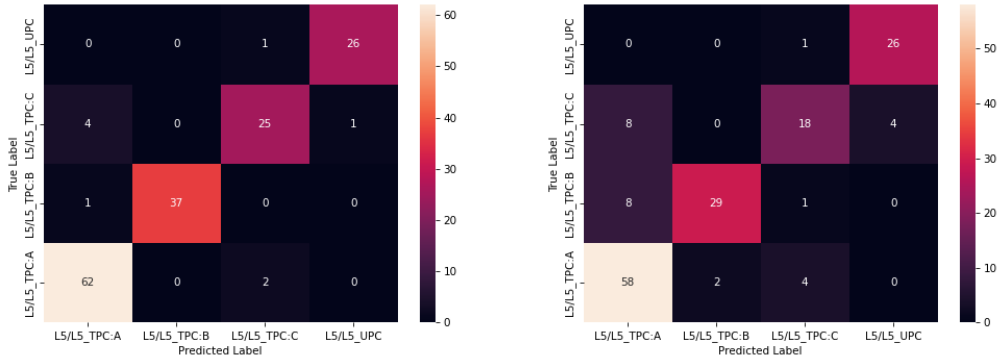
The `MultiConcat` model using all available embeddings (GNN, PI-CNN, B, S, SW, L, M)[28] performed L5 classification with approximately 80% accuracy or better across all tested embedding dimensions. The embedding dimension that resulted in the best accuracy was 16 (0.89 ± 0.05); both dimensions 16 and 32 (0.85 ± 0.06) performed substantially better than the reproduced GNN benchmark (0.82 ± 0.07). Table 3 summarizes these results.

Table 3: Accuracy of `MultiConcat` (all embeddings) Across Embedding Dimensions Compared to `ManNet` (Reproduction) in *Experiment 3.1*

Model	Accuracy
MultiConcat (emb. dim. = 8)	0.81 ± 0.09
MultiConcat (emb. dim. = 16)	$\mathbf{0.89 \pm 0.05}$
MultiConcat (emb. dim. = 32)	0.86 ± 0.06
MultiConcat (emb. dim. = 128)	0.84 ± 0.05
MultiConcat (emb. dim. = 512)	0.79 ± 0.05
ManNet (reproduction)	0.82 ± 0.07

4.3.2) *Experiment 3.2*: Embedding Subsets

The highest accuracy of 94% ± 0.06 was reached by finetuning on the subset [GNN, B, S, L, M]. Without finetuning, various combinations of embeddings reached an accuracy of approximately 93%. These combinations differed only in standard deviation: [GNN, B, SW, L, M] (±7%), [GNN, B, S, L, M] (±6%), and [W, L, M] (±5%).



(a) Finetuned Multi-Embedding Fusion Model `MultiConcat` (GNN, B, SW, LS).

(b) Reproduction of L5 PC classification with `ManNet` [7].

Figure 14: Comparison of the confusion matrix of the best `MultiConcat` model from *Experiment 3.2* with the ManNet benchmark on Janelia-L5-PCs.

[28] Embedding abbreviation descriptions are given in Table 5

Comparing the best results achieved by the `MultiConcat` model with those from Kanari et al. [7], the Multi-Embedding Fusion model improves classification accuracy on the Janelia-L5-PC dataset by 12% with respect to the reproduction and by 13% compared to the accuracy reported in the original paper. Figure 14a and Figure 14b compare the confusion matrices with respect to the reproduction.

The different subset selection tests were conducted with embedding dimensions 16 and 32 following the results from *Experiment 3.1.* Combinations with many (more than five) embeddings mostly tended toward dimension 16, while most others performed better with dimension 32. Since more combinations reached peak performance with 32-dimensional embeddings, *Experiment 3.2* was conducted at embedding dimension 32.

In a first step, pairwise and single embedding combinations were inspected. It was observed that Persistent Landscape is the best embedding (89% ± 7%), followed by Bottleneck Distance (84% ± 7%). All embeddings were shown to be effective (approximately 82%+), except sliced Wasserstein and morphometrics (approximately 70%). For the vast majority of embedding combinations, pairing with another embedding improved classification by at least one percent relative to the better embedding of the pair (see Figure 15). The only exceptions are the combination [PI-CNN, LS] and combinations including underperforming embeddings (SW, M).

Note: *Again, no parameter optimization was done for this indicative table Figure 15. Pairwise embeddings were trained for 20 epochs due to generally fast convergence and for efficiency; however, for combinations where slow convergence was observed, classification was repeated at 30 epochs, and the latter result was reported if it was an improvement.*

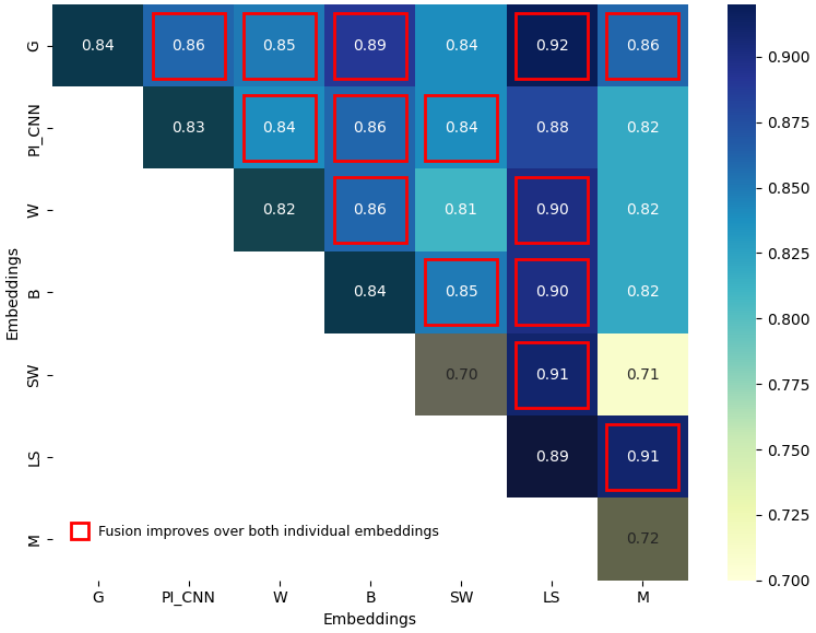


Figure 15: Pairwise accuracies using two-embedding training, highlighting combinations that outperform both individual embeddings.

Based on these initial results, different subsets of embeddings were tested on an exploratory basis. The combinations were trained with varying epochs and learning rates, as the speed of convergence varied among the different groups. Table 4[29] presents a selection of results, grouped by embedding origin.

[29]Embedding abbreviation descriptions are given in Table 5

Table 4: Selection of Embedding Combinations Grouped by Embedding Origin (emb. dim. = 32) for *Experiment 3.2*

Embedding Origin	Embeddings	Accuracy (±SD)
Graph, TMD, Stats	GNN, B, SW, LS, M	**0.93 ± 0.07**
	GNN, B, LS, M	0.92 ± 0.06
	GNN, W, SW, LS, M	0.92 ± 0.07
	GNN, LS, M	0.91 ± 0.06
	GNN, CNN-PI, LS, M	0.89 ± 0.04
	GNN, CNN-PI, B, W, SW, LS, M	0.86 ± 0.06
Graph, TMD	GNN, B, LS	**0.93 ± 0.06**
	GNN, B, W	0.91 ± 0.03
	GNN, CNN-PI	0.87 ± 0.06
Graph, Stats	GNN, M	0.86 ± 0.09
TMD, Stats	W, LS, M	**0.93 ± 0.05**
	B, LS, M	0.89 ± 0.10
	CNN-PI, SW, M	0.77 ± 0.07
TMD	B, W, SW, LS	0.91 ± 0.07
	B, W, LS	0.89 ± 0.08
	CNN-PI, W, SW, LS, B	0.89 ± 0.05
Graph (Benchmark)	ManNet (reproduction)	0.82 ± 0.07

4.3.3) *Experiment 3.3*: Embedding Preference

Qualitative analysis shows that the learned scalar weights per embedding do not exhibit a clear preference for any particular embedding. The selection weights fluctuate around a uniform distribution and never diverge significantly. Different cross-validation runs with the same model show some variation in top pick preferences, but these differences remain small. Generally, the weight assigned to an embedding X was slightly lower when the model based on embeddings (A, B) outperformed the model based on embeddings (A, B, X), which is an expected result.

5 Discussion

As outlined in Section 3.4.1), the motivation for this work stems from the observation that descriptors from distinct feature origins—Morphometrics [7, 26], TMD-based topological summaries [7, 6], and Graph-based embeddings [30, 31]—each demonstrate strong classification performance in isolation, as shown in prior studies [7, 6, 26] and elaborated in Section 2.3.2). *Experiments 1 & 2* reaffirm some of these findings, supporting the utility of these modalities for neuronal morphology classification. Each embedding class captures a unique perspective: Morphometrics encode global shape statistics [24, 26]; TMD-based methods capture multiscale topological structures based on persistent homology [4, 6]; and GNN-based graph embeddings [7, 30, 31] extract rich structural patterns from 3D graphs.

This diversity suggests the presence of complementary information across modalities. Thus, a central hypothesis of *Experiment 3* was that combining embeddings from different modalities could lead to enhanced classification performance. The proposed Multi-Modal Embedding Fusion approach was designed to explore this hypothesis.

To evaluate both the complementarity and the standalone strength of each embedding, I conducted an exhaustive comparison of single and pairwise combinations (Figure 15). This allowed for a deeper understanding of how representations from different sources interact. Based on these insights, more complex fusion models incorporating multiple embeddings were then explored.

5.1 Dimensionality

Lower-dimensional embeddings were generally favored, especially in multi-modal settings. Combining numerous large embeddings leads to a high-dimensional feature space, which increases the risk of overfitting—particularly on smaller datasets [36] such as *Pyramidal Cells* from Janelia [23] used in *Experiment 3*. These observations suggest that embedding dimensionality must be carefully balanced with model complexity.

5.2 Single Embedding

When evaluating the MultiConcat network using single embeddings (i.e., without fusion), Persistence Landscapes (LS) and Bottleneck (B) stood out as particularly strong. This suggests that some TDA-based vectorization methods, when coupled with an MLP classifier, already possess high descriptive power—outperforming even the GNN-based ManNet benchmark from Kanari et al. [7]. This underlines the effectiveness of TMD-derived features for morphology classification.

Interestingly, the standalone GNN model [30] achieved an accuracy of 0.84, outperforming the ManNet benchmark (0.82) [7]. This is attributed to a reduced embedding dimensionality (2nd ChebConv [30, 35] output reduced to 32, compared to 512 in ManNet) as required for matching the uniform embedding dimension, which was chosen to be 32. This likely decreased overfitting due to a large feature space [36]. Repeating the ManNet experiment with these modified dimensions confirms this: a performance of 0.84 was achieved using the default hyperparameter of *Experiment 3* (lr: 0.001). Keeping the default learning rate from *Experiment 2* (lr: 0.005), the result even increased to 0.85—both matching or surpassing our isolated GNN result via CLI (subsubsection 3.3.2)). This provides further support for the dimensionality analysis even in the single-embedding regime.

5.3 Pairwise Fusion

Pairwise fusion of the available embeddings provides a first indicator of complementarity as well as the performance of the fusion model. Its results demonstrate a consistent pattern: most combinations outperform their respective single embeddings, indicating that meaningful complementary information is indeed present. Noteworthy examples include GNN + B, which improved accuracy by 5%, and GNN + LS, which added 3% over LS alone—an especially meaningful gain given LS's already high baseline. This indicates that both the mentioned TMD-based embeddings and the graph-based embedding likely complement each other well.

Conversely, a few combinations—particularly those involving SW or M—performed worse than their strongest individual component. This outcome is not unexpected: when pairing two embeddings, if one

has substantially lower representational quality, the combined performance may not improve and can even degrade. In such cases, the weaker embedding may act as noise, diminishing the effectiveness of the stronger one [37, 38].

5.4 Multi Fusion

Larger combinations yielded several top-performing models, including [GNN, B, LS, M] and [W, LS, M], both achieving accuracies of approximately 0.93. These models indicate that effective fusion does not require stacking all available features, but rather selecting a well-balanced subset that spans multiple modalities.

Across these larger combinations, the most consistent observation was that top-performing models (93% +) always included at least three embeddings from at least two distinct modalities, supporting the hypothesis that structural, topological, and morphometric features offer complementary perspectives. However, no clear rule for a well-performing subset is identifiable.

Another consistent trend is the inclusion of LS in most top-performing models, reaffirming its strong individual performance and its value in fusion. The ManNet benchmark [7] (0.82 respectively 0.85 with refined dimensionality) was surpassed by most multi-modal combinations created based on the insight from pairwise fusion. This highlights the added value from multi-embedding fusion.

Furthermore, classification success has been shown not to be about stacking as many features as possible, but about selecting a synergistic subset. Fusing complementary modalities leads to more robust and generalizable performance, while unbalanced combinations may dilute signal or introduce noise.

Note: *Experiment 3.3 offers limited insight. The learned parameters exhibit random fluctuations, suggesting that either all vectorizations contribute similarly to the embedding fusion—supporting the idea that complementary information exists across embeddings—or that the weight updates during training were too small to produce noticeable changes.*

5.5 Generalization

As discussed previously, embedding dimensionality was found to be a critical factor. Smaller embedding sizes often yielded better performance—particularly in GNN-based settings—likely due to reduced overfitting on this relatively small dataset. The same was observed for variable hidden dimensions, where best performing setups included small-sized hidden dimensions on the order of magnitude of the embedding dimension, indicating that mapping to a higher hidden space introduces noise rather than being beneficial, likely also a cause of the small dataset at hand[36].

Apart from embedding dimensionality, key hyperparameters that impact generalization included the learning rate and number of training epochs. Most models converged between 20–30 epochs, though some combinations benefited from longer training (up to 50 epochs), while others generalized better with shorter runs (e.g., 12 epochs) and overfitted during longer training[36]. This variability likely reflects differences in fusion complexity and convergence dynamics across embedding types.

5.6 Limitations and Outlook

While this study focuses on a single cortical layer, the promising results suggest potential for broader applicability. Still, extensive testing across brain regions and datasets is required to validate universality.

Additionally, further refinement of the fusion mechanism—beyond feature concatenation—may yield even stronger results. Alternatives to concatenation include different aggregation functions such as averaging or using an MLP for aggregation [39]. Alternatively, other ensemble architectures which differ from pre-classification head concatenation could be examined for this task[40].

6 Conclusion

This project began with the theoretical groundwork of understanding **Persistent Homology**[2], a fundamental concept in **Algebraic Topology** that plays a crucial role in **Topological Data Analysis (TDA)**[13, 14]. I developed the necessary mathematical framework for understanding persistent homology based on prerequisites appropriate for an engineering graduate student without a strong background in mathematics.

The self-contained theoretical section guided my own process of understanding the foundations of TDA and summarizes the essential concepts for readers with a similar background who are interested in learning about the theoretical background of Topological Data Analysis from scratch.

Having established this foundation, the project explored how the TMD algorithm [4]can be used to derive topological descriptions of a space, and how such topological features can be vectorized using descriptors such as persistence landscapes, persistence images, and pairwise distance matrices based on distances such as Wasserstein.

Building on this theoretical base, I reproduced key results from the literature [7, 6]to validate both my implementation and the practical utility of TMD-based descriptors in classifying neuronal morphologies.

With the goal of exploring deep learning methods such as Graph Convolutional Neural Networks (GNN) [30, 31] for morphology classification, the project then examined the complementarity of different feature representations, including morphometrics, topological summaries, and graph-based embeddings discussed in Kanari et al. (2024) [7]. Through systematic experimentation with individual embeddings, pairwise combinations, and full multimodal fusion, I demonstrated that Multi-Embedding Fusion consistently improves classification accuracy on selected datasets (*L5 Pyramidal Cells from the Janelia Laboratory*[23]). This confirmed the hypothesis that different embedding modalities capture complementary, yet informative, aspects of neuronal structure.

Notably, topological embeddings—particularly persistence landscapes—proved highly effective in multi-embedding fusion, both independently and in combination with GNN-embedded or morphometric features. The experiments also showcased the importance of feature dimensionality and generalization when training deep classifiers on relatively small datasets [36].

The key contributions of this project are summarized as follows:

- Conducting a comprehensive literature review and constructing the theoretical framework for Topological Data Analysis (TDA)[13, 14], and the Topological Morphology Descriptor algorithm (TMD)[4].
- Confirming the usefulness of TMD-based methods for neuronal morphology classification.[6]
- Exploring various morphology representation modalities and evaluating their utility in (deep) machine learning applications[7].
- Proposing and implementing a Multi-Embedding Fusion Model, leveraging the complementary nature of different embedding modalities.
- Successfully improving the benchmark classification accuracy on the *Janelia L5 Pyramidal Cells Dataset* from 81% (as reported by Kanari et al. [7]) to 94% using the proposed MultiConcat model.

Ultimately, this project presents a unified and extensible framework for neuronal morphology classification. By leveraging the unique strengths of diverse embedding types through pre-classification head embedding fusion, it achieves superior performance over established baselines on the selected dataset, providing a valid starting point for improving classification of neural morphologies.

7 Code Availability

The code used for this project is available via the repository `algTopo-ML-neuronCls` on GitHub at: github.com/janzgraggen/algTopo-ML-neuronCls

References

[1] A. Hatcher, *Algebraic Topology*. Cambridge, England: Cambridge University Press, 2002.

[2] H. Edelsbrunner and D. Morozov, "Persistent homology: Theory and practice," in *European Congress of Mathematics*. European Mathematical Society Publishing House, 2014, pp. 31–50.

[3] R. Ghrist, "Barcodes: The persistent topology of data," *Bulletin of the American Mathematical Society (New Series)*, vol. 45, no. 1, pp. 61–75, January 2008, article electronically published on October 26, 2007.

[4] L. Kanari, P. Dłotko, M. Scolamiero, R. Levi, J. Shillcock, K. Hess, and H. Markram, "A topological representation of branching neuronal morphologies," *Neuroinformatics*, vol. 16, no. 1, pp. 3–13, Jan 2018.

[5] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier, "Persistence images: A stable vector representation of persistent homology," *Journal of Machine Learning Research*, vol. 18, no. 8, pp. 1–35, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-337.html

[6] L. Kanari, S. Ramaswamy, Y. Shi, S. Morand, J. Meystre, R. Perin, M. Abdellah, Y. Wang, K. Hess, and H. Markram, "Objective morphological classification of neocortical pyramidal cells," *Cerebral Cortex*, vol. 29, pp. 1719–1735, April 2019, advance Access Publication Date: 31 January 2019.

[7] L. Kanari, S. Schmidt, F. Casalegno, E. Delattre, J. Banjac, T. Negrello, Y. Shi, J. Meystre, M. Defferrard, F. Schürmann, and H. Markram, "Deep learning for classifying neuronal morphologies: combining topological data analysis and graph neural networks," *bioRxiv*, September 2024, preprint, not peer-reviewed. [Online]. Available: https://doi.org/10.1101/2024.09.13.612635

[8] N. Jacobson, *Basic Algebra I*, 2nd ed. Mineola, New York: Dover Publications, 2009, originally published by W. H. Freeman and Company, San Francisco, in 1985.

[9] J. R. Munkres, *Topology*, 2nd ed. Harlow, Essex, England: Pearson Education Limited, 2014.

[10] J. Whitehead, "Combinatorial homotopy. i," *Bulletin of the American Mathematical Society*, vol. 55, no. 3, pp. 213–245, 1949.

[11] K. Jänich, *Topology*. Springer-Verlag, 1984.

[12] M. A. Armstrong, *Basic Topology*. Springer-Verlag, 1983.

[13] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society (New Series)*, vol. 46, no. 2, pp. 255–308, 2009.

[14] ——, "Topological pattern recognition for point cloud data," *Acta Numerica*, pp. 289–368, 2014.

[15] P. Bubenik and P. Dłotko, "A persistence landscapes toolbox for topological statistics," *Journal of Symbolic Computation*, vol. 78, pp. 91–114, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747717116300112

[16] M. Carrière, M. Cuturi, and S. Oudot, "Sliced wasserstein kernel for persistence diagrams," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, August 2017, pp. 1–10.

[17] Y. Deitcher, G. Eyal, L. Kanari, M. B. Verhoog, G. A. A. Kahou, H. D. Mansvelder, C. P. J. de Kock, and I. Segev, "Comprehensive morpho-electrotonic analysis shows 2 distinct classes of l2 and l3 pyramidal neurons in human temporal cortex," *Cerebral Cortex*, vol. 27, no. 11, pp. 5398–5414, 2017.

[18] N. Spruston, "Pyramidal neurons: Dendritic structure and synaptic integration," *Nature Reviews Neuroscience*, vol. 9, no. 3, pp. 206–221, 2008.

[19] S. Ramaswamy and H. Markram, "Anatomy and physiology of the thick-tufted layer 5 pyramidal neuron," *Frontiers in Cellular Neuroscience*, vol. 9, p. 233, 2015.

[20] H. Markram, E. Muller, S. Ramaswamy, M. W. Reimann, and et al., "Reconstruction and simulation of neocortical microcircuitry," *Cell*, vol. 163, pp. 456–492, 2015.

[21] J. DeFelipe and I. Fariñas, "The pyramidal neuron of the cerebral cortex: Morphological and chemical characteristics of the synaptic inputs," *Progress in Neurobiology*, vol. 39, no. 6, pp. 563–607, 1992.

[22] V. Zinchenko, J. Hugger, V. Uhlmann, D. Arendt, and A. Kreshuk, "Morphofeatures for unsupervised exploration of cell types, tissues, and organs in volume electron microscopy," vol. 12, p. e80918, feb 2023. [Online]. Available: https://doi.org/10.7554/eLife.80918

[23] J. Winnubst, E. Bas, T. A. Ferreira, Z. Wu, M. N. Economo, P. Edson, B. J. Arthur, C. Bruns, K. Rokicki, D. M. Schauder, D. J. Olbris, S. D. Murphy, D. Ackerman, C. Arshadi, P. Baldwin, R. Blake, A. Elsayed, M. Hasan, D. Ramirez, B. Paiva dos Santos, M. Weldon, A. Zafar, J. T. Dudman, C. R. Gerfen, A. W. Hantman, W. L. Korff, S. M. Sternson, N. Spruston, K. Svoboda, and J. Chandrashekar, "Reconstruction of 1,000 projection neurons reveals new cell types and organization of long-range connectivity in the mouse brain," *Cell*, vol. 179, pp. 268–281.e13, 2019.

[24] S. Laturnus, D. Kobak, and P. Berens, "A systematic evaluation of interneuron morphology representations for cell type discrimination," *Neuroinformatics*, vol. 18, pp. 591–609, 2020. [Online]. Available: https://doi.org/10.1007/s12021-020-09461-z

[25] R. Scorcioni, S. Polavaram, and G. A. Ascoli, "L-measure: A web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies," *Nature Protocols*, vol. 3, pp. 866–876, 2008.

[26] G. A. Ascoli, D. E. Donohue, and M. Halavi, "Neuromorpho.org: A central resource for neuronal morphologies," *The Journal of Neuroscience*, vol. 27, pp. 9247–9251, 2007.

[27] L. Kanari, H. Dictus, A. Chalimourda, W. Van Geit, B. Coste, J. Shillcock, K. Hess, and H. Markram, "Computational synthesis of cortical dendritic morphologies," 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/04/17/2020.04.15.040410

[28] S. Muralidhar, Y. Wang, and H. Markram, "Synaptic and cellular organization of layer 1 of the developing rat somatosensory cortex," *Frontiers in Neuroanatomy*, vol. 7, 2013.

[29] S. Morand, "tda_toolbox," https://github.com/Eagleseb/tda_toolbox, 2024, accessed: 2025-04.

[30] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf

[31] W. L. Hamilton, *Graph Representation Learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020, vol. 14, no. 3.

[32] N. W. Gouwens, S. Sorensen, J. Berg, C.-Y. Lee, T. Jarsky, J. Ting, S. Sunkin, D. Feng, C. A. Anastassiou, E. Barkan, and et al., "Classification of electrophysiological and morphological neuron types in the mouse visual cortex," *Nature Neuroscience*, vol. 22, pp. 1182–1195, 2019.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019, pp. 8024–8035.

[35] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[36] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, p. 022022, 2019.

[37] X. Zhang, Q. Liu, X. Wang, Z. Wu, and Y. Xie, "Redundancy is not what you need: An embedding fusion graph auto-encoder for self-supervised graph representation learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2024, available at: https://pubmed.ncbi.nlm.nih.gov/38300769.

[38] T. Hoang, J. Singh, D. Lee, and T. Nguyen, "The best of both worlds: Combining learned embeddings with engineered features for accurate prediction of correct patches," *Proceedings of the ACM/IEEE International Conference on Automated Software Engineering (ASE)*, 2023. [Online]. Available: https://dl.acm.org/doi/full/10.1145/3576039

[39] J. Martinez, "Embeddings merge 101: A practical guide to merging embeddings," August 2024, accessed: 2025-05-30. [Online]. Available: https://medium.com/mantisnlp/how-to-combine-several-embeddings-models-8e7bc9a00330

[40] K.-H. Chan, S.-K. Im, and W. Ke, "A multiple classifier approach for concatenate-designed neural networks," 01 2021.

A Apendix

A.1 Embedding Abbreviations

Table 5: Embedding Abbreviations

Abbrev.	Description
GNN	Graph Neural Network embedding, based on GraphEmbedder form Morphoclass [7]
CNN-PI	Convolutional Neural Network for Persistence Images, based on CNNEmbedder form Morphoclass[7]
B	Bottleneck Distance embedding, based on pairwise distance matrix
W	Wasserstein Distance embedding, based on pairwise distance matrix
SW	Sliced Wasserstein Distance embedding, based on pairwise distance matrix
LS	Persistence Landscape embedding
M	Morphometrics Embbeding

A.2 Morphometrics Features per Neutrite Type

Table 6: List of Morphometric Features per Neurite Type (Basal and Apical Dendrites and Axons)

Feature	Feature	Feature
mean local bifurcation angles	mean section radial distances	mean number of bifurcations
max section Strahler orders	mean number of leaves	mean section Strahler orders
mean number of neurites	min section term lengths	max section term lengths
mean number of sections	mean section term lengths	mean partition asymmetry
mean partition asymmetry length	mean section term radial distances	mean section tortuosity
mean section volumes	mean remote bifurcation angles	mean segment meander angles
mean terminal path lengths	mean section areas	mean total area
min section bif lengths	max section bif lengths	sum total area
mean section bif lengths	min total area per neurite	max total area per neurite
mean section bif radial distances	min section branch orders	mean total area per neurite
max section branch orders	mean section branch orders	sum total length
mean section end distances	min total length per neurite	max total length per neurite
mean section lengths	mean total length per neurite	min section lengths
max section lengths	mean total volume	sum total volume
min section path distances	max total volume per neurite	mean section path distances
median section path distances	min total volume per neurite	mean total volume per neurite

A.3 Morphometrics Features Global

Table 7: List of Morphometrics per Neuron (Global Features)

Feature	**Feature**
mean max radial distance	mean number of neurites
mean number of sections per neurite	mean soma radius
mean soma surface area	mean soma volume
mean total area per neurite	mean total depth
mean total height	mean total length per neurite
mean total volume per neurite	mean total width