RESEARCH ARTICLE

Metrics for comparing neuronal tree shapes based on persistent homology

Yanjie Li[1], **Dingkang Wang**[1], **Giorgio A. Ascoli**[2], **Partha Mitra**[3], **Yusu Wang**[1]*

1 Computer Science and Engineering Department, The Ohio State University, Columbus, OH 43221, United States of America, **2** Krasnow Institute for Advanced Study, George Mason University, Fairfax, VA 22030, United States of America, **3** Cold Spring Harbor Laboratory, Cold Spring Harbor, NY 11724, United States of America

* yusu@cse.ohio-state.edu

Citation: Li Y, Wang D, Ascoli GA, Mitra P, Wang Y (2017) Metrics for comparing neuronal tree shapes based on persistent homology. PLoS ONE 12(8): e0182184. https://doi.org/10.1371/journal.pone.0182184

Editor: Gennady Cymbalyuk, Georgia State University, UNITED STATES

Received: March 6, 2017

Accepted: July 13, 2017

Published: August 15, 2017

Data Availability Statement: All data / software tool are available from GitHub at https://github.com/Nevermore520/NeuronTools. See directory ./Experiments for all data used in this submission.

Funding: This work was supported by The Crick-Clay Professorship at Cold Spring Harbor Lab to Dr. Mitra, National Institute of Health grant NIH 1R01EB022899-01, National Institute of Health grant NIH R01NS039600, National Institute of Health grant NIH R01NS086082, National Science Foundation grant BRAIN Eager 1450957, and National Science Foundation grant BRAIN Eager

Abstract

As more and more neuroanatomical data are made available through efforts such as NeuroMorpho.Org and FlyCircuit.org, the need to develop computational tools to facilitate automatic knowledge discovery from such large datasets becomes more urgent. One fundamental question is how best to compare neuron structures, for instance to organize and classify large collection of neurons. We aim to develop a flexible yet powerful framework to support comparison and classification of large collection of neuron structures efficiently. Specifically we propose to use a topological persistence-based feature vectorization framework. Existing methods to vectorize a neuron (i.e, convert a neuron to a feature vector so as to support efficient comparison and/or searching) typically rely on statistics or summaries of morphometric information, such as the average or maximum local torque angle or partition asymmetry. These simple summaries have limited power in encoding global tree structures. Based on the concept of topological persistence recently developed in the field of computational topology, we vectorize each neuron structure into a simple yet informative summary. In particular, each type of information of interest can be represented as a descriptor function defined on the neuron tree, which is then mapped to a simple persistence-signature. Our framework can encode both local and global tree structure, as well as other information of interest (electrophysiological or dynamical measures), by considering multiple descriptor functions on the neuron. The resulting persistence-based signature is potentially more informative than simple statistical summaries (such as average/mean/max) of morphometric quantities—Indeed, we show that using a certain descriptor function will give a persistence-based signature containing strictly more information than the classical Sholl analysis. At the same time, our framework retains the efficiency associated with treating neurons as points in a simple Euclidean feature space, which would be important for constructing efficient searching or indexing structures over them. We present preliminary experimental results to demonstrate the effectiveness of our persistence-based neuronal feature vectorization framework.

1546335. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Introduction

Neuronal cells have a unique geometrical characteristic: tree-like axonal and dendritic processes that can be many orders of magnitude bigger than the cell bodies (somata). These dendritic and axonal trees are fundamental to the operation of neurons, since they enable the coordinated long distance communication of electrical signals, and also enable the complex short and long distance connectivity architecture that is central to nervous system function. In analyzing the circuit properties, a data reduction is often made to a connectivity matrix (synaptic connections or mesoscale regional connections), without taking into account the neuronal geometry or topology *per se*. However, it is highly likely that the neuronal geometry plays a critical role in determining the capabilities of the circuit—the geometry is intimately tied to the timing properties of signals in the nervous system and also determines the algorithmic capabilities of the spatially extended circuitry. Since the nervous system enables rapid responses to environmental stimuli to govern behavior, time is of essence. The spatial relations between different inputs to a dendritic tree are important for how the corresponding signals integrate. The tree geometries of neuronal processes reflect developmental dynamics, including the growth and pruning of these processes.

Despite the importance of the geometrical and topological properties of neuronal trees, the characterization and analysis of these properties pose conceptual and methodological challenges. A basic reason for this is that the tree geometries are not naturally characterized by points in some suitable vector space. For tree shapes to be vectors, one should be able to add and subtract tree shapes. There is no natural way to do this. Since vectors spaces (and linear algebra) are fundamental to the data analysis techniques that are widely used, this poses a conundrum. One way out is to map the neuronal geometries to a vector space (through a suitable choice of feature vectors); however, this entails loss of information in a potentially *ad hoc* manner. The alternative is to use a mathematical description that is more naturally suited to tree shapes.

One possibility is to characterize neuronal trees as points in a metric space, where distances between objects are defined, but addition and subtraction of objects need not be defined. While not widely used, metric space techniques do have precedence in neuronal data analysis (e.g. metric space methods for spike trains). Central to such analyses is a suitable choice of metric or distance between trees. One way to achieve this is to first embed the trees into a vector space, then use a metric in that vector space. However, this intermediate vector space representation could obscure the study of structures that might be present purely in the metric space framework, and also requires the *ad hoc* choice of a vector space representation, so it does not address the basic issue.

In this paper we explore the possibility of directly defining the associated metric space by exploring different tree-metrics, and propose a comprehensive methodology that can also deal with the development or growth of neuron trees and dynamics defined on the trees. The methods may involve reduction to vectors at an intermediate point of analysis, but this happens in a natural and controlled way without ad hoc feature selection. It is also possible to proceed without reduction to vectors, which we indicate but do not pursue in detail in this manuscript. We rely on techniques developed over the last decade based on ideas of topological persistence, that have gained widespread use outside in other applications dealing with geometrical and topological data analysis.

An important application (though not the only one) is to the problem of classifying neurons into classes or types. Axonal or dendritic morphology has been used from early days (cf. Cajal) for such classification purposes, and has been one of the major motivators for past quantitative work based on intermediate feature vector representations. The introduction of computational geometry and topology techniques to this data analysis problem brings in a modern toolkit, that is also well suited to the large data sets that are becoming available through efforts such as NeuroMorpho.Org [1] and FlyCircuit.org [2]. A central question for these data sets is how best to compare neuron structures. This is needed to organize and classify large collections of neurons, to understand variability within a cell type, and to identify features that distinguish neurons. Despite extensive attention from researchers, this problem remains challenging [3]. A broad spectrum of methods to compare neuronal geometries have been developed in this big data context. On one end of the methodological spectrum, the aim is to develop efficient similarity / distance measures for neurons to facilitate efficient classification, search and indexing of neuron data, or as a way to characterize key features of neuron structures. On the other end of the methodological spectrum, the aim is to find a detailed alignment (correspondence) between two or multiple neuron trees to help understand similarity and variation among structures in detail, to help construct consensus or mean structure, and so on. There is typically a trade-off of efficiency versus sensitivity (to structure variation) as we move from one end to the other end of the methodological spectrum.

In this paper, we focus on the efficient end of the spectrum of methods, and aim to develop a flexible yet powerful framework to compare large collection of neuron structures efficiently, while bringing in modern tools for computational geometry and topology.

Related work

On the efficient end of the method spectrum, there are a family of what one might call *feature-vectorization* methods. Such methods map each neuron structure into a point (a feature vector) in an investigator-defined feature space (often Euclidean space) and the distance between two neurons is measured by the computationally friendly L_p-norm between their corresponding feature vectors. Then one can leverage the large literature on searching, nearest-neighbor queries, clustering and classification under L_p-norms, to facilitate efficient automatic classification as well as indexing /querying in a big database of neuron structures. One popular way to vectorize a neuron structure is to map it to features consisting of a subset of summarizing morphometric parameters (such as average / max local torque angles) as computed by the L-Measure tool [4]; see e.g, [5, 6, 7]. It has also been observed [3] that classic Sholl-like analysis [8], which counts the number of intersections between neuronal tree with concentric spherical shells centered at soma, provide effective measurements for neuron classification [9, 10, 11]. Other approaches in this family include mapping the skeleton of neuron structure to a density field [12], or representing a neuron by a collection of segments (each represented as a vector) as used in NBLAST [13].

In contrast, on the other end of the method spectrum, at the most sensitive (discriminative) level, one aims to establish (complete or partial) alignments / correspondences between two or multiple neuronal trees, so as to help understand similarity and variation among structures in detail, and to construct a consensus or mean structure. The importance of the specific branching pattern and the tree shape of neurons in their functionality has long been recognized [3]. The neuron structures can be treated as combinatorial trees (where only the connection pattern between nodes matter) or as geometric trees (where locations of nodes and geometric shapes of arcs are also considered). Methods in this category often aim to find correspondences between two (neuron) trees, as well as to develop a tree distance to measure the quality of the resulting tree alignment. One important development in this direction is the use of a tree edit distance (TED) for aligning neuron trees [14, 15]. The tree edit distance can be considered as an extension of the string-edit distance. It measures the distance between two trees by identifying the minimum cost sequence of "edit" operations to convert one tree to the other tree. It is a natural distance for comparing trees, and has been used in various biological applications, such as for comparing phylogenetic trees. Unfortunately, the tree edit distance is NP hard to compute [16], or even to approximate [17]. So current applications use a constrained TED, which can be solved by dynamic programming in polynomial time. The constraints require that ancestor/descendant relations be preserved by the correspondences [14, 15]. The original constrained TED does not model the shape of tree branches, though the alignment used by the multiscale neuron comparison and classification tool BlastNeuron considers the shape of branches to some extent [18]. The DIADEM metric [19] presents a more detailed alignment targeted to the special case of comparing a reconstructed neuron structure with a "*gold standard*" structure.

In the middle of the spectrum are methods of varying sensitivity and computational costs. Path2Path [20] converts a neuron tree into a set of paths (curves) and then measures distance between two neurons by the distance between corresponding sets of curves. This approach helps to take branch shape into account, but the tree combinatorial structure is somewhat lost. A more enriched model [21] represents a tree as a main curve with several branches (and possibly sub-branches), and uses a dynamic time warping algorithm to align these branches along the main curve. Recognizing the importance of locality of neuronal arborisations, Zhao and Plaza [22] converts neurons into one dimensional distributions of branching density for comparison. Finally, in an interesting recent development [23], Gillette et al. encode the combinatorial structure of a tree as a sequence and compares two or multiple neuron structures using the large literature on sequence alignments.

New method

In this paper, we focus on the efficient end of the spectrum. We note that current methods to vectorize a neuron typically rely on statistics or summaries of important morphometric information, such as the average or maximum local torque angle or partition asymmetry. These simple summaries have limited power in encoding global tree structures. We leverage recent developments in topological data analysis [24, 25, 26], especially in persistent homology [27, 28, 29], and propose a new persistence-based feature vectorization framework, which have advantages over previous approaches. First, it provides a unified general framework that can encompass a variety of properties associated with neurons, both static and dynamic. Specifically, each property of interest can be represented as a *descriptor function* defined on the neuron tree, which is then mapped to a simple persistence-signature. This procedure is repeated with other descriptor functions, and the collection of these signatures is considered together as a feature vector. As a result, our framework can encode both local and global tree structure, as well as other information of interest (that pertain to dynamical and electrophysiological properties of neurons), by considering a suitable set of descriptor functions. The resulting persistence-based signature is geometrically meaningful and more informative than simple statistical summaries (such as average, sum, or max) of morphometric quantities. As an example, in Section *Materials and methods*, we show that by using a natural descriptor function in our framework, our persistence signature is in a mathematically precise sense more informative than the classical Sholl analysis [8]. Secondly, by vectorizing the persistence information in a natural manner, our framework retains the efficiency associated with treating neurons as points in a simple Euclidean feature space. We present some preliminary experimental results to demonstrate the effectiveness of our proposed framework. Third, the method generalizes to neuronal shapes that change over time (due to development or experience dependent plasticity), and therefore provides a natural method to capture developmental dynamics.

Note concering contemporaneous work: During the course of preparing this manuscript, we were made aware of independent work published on the arXiv [30], developed by Kanari et al. Similar to our paper, this paper also proposes to use topological persistence-based profiles to compare neuron morphologies. We point out that these two lines of work, despite their similarity, were developed independently. A preliminary presentation of our work was made in poster form at the US BRAIN Initiative annual meeting in *December 2015* in Washington DC. We would like to note that using persistence-based metrics to analyze geometrical graphs or trees have been used in the prior literature (see e.g, [31] for graphs and [32] for analyzing brain artery trees), and do not constitute novel elements in either our work or in the preprint by Kanari et al, but are applications of these literature ideas to neuronal trees. However, our respective applications differ in detail. We use a different way to compute persistence-based feature vectors and their distances. We formulate and prove that the persistence-based signature derived from the Euclidean distance function is strictly more informative than the typical information used in the classical Sholl analysis [8]. We also discuss how to integrate multiple descriptor functions, and provide a more detailed roadmap based on our approach suitable for the study of electrophysiological and developmental dynamics. Our experimental results are based on using the geodesic distance function as the descriptor function; while results based on the radial distance function from the root (referred to as Euclidean distance function in our paper) were reported in [30] (it is pointed out in [30] that their method can be applied to other descriptor functions as well). (In our experiments, we observe that geodesic distance function in general achieves better performance than the Euclidean distance function; see S2 File.) We also report comparison of persistence-based feature vectors with Sholl analysis as well as with L-Measure quantities in our experiments. We publicly release the persistence feature-vectorization as well as the neuron-tree comparison software.

Materials and methods

We develop a persistence-based signature for neuron structures. Specifically, we model a single neuron as a geometric tree $T \subset \mathbb{R}^3$ embedded in the three-dimensional Euclidean space $\mathbb{R}^3$, where arcs connecting tree nodes are modeled as (polygonal) curves. To incorporate various information of interests on the neuron trees, we model them as *descriptor functions* defined on T. We then apply the so-called *topological persistence* to summarize these descriptor functions, to map an input neuron tree (together with various structural or biochemical information on it) into a signature (feature vector). The high-level pipeline is shown in Fig 1—here for simplicity, we use a single descriptor function as an example. But as we describe later, this framework can be extended to multiple descriptor functions.

Step 1. Persistence diagram summary

Persistent homology [27] is a basic methodology to characterize and summarize shapes and functions, as well as to identify meaningful features and separate them from "noise" [24, 25]. The underlying space X is examined using a mathematical construct called a *filtration*. A filtration of the space X consists of a nested sequence of indexed subsets of X with the index chosen from an ordered set (such as the set of integers or the set of real numbers), e.g. $X_1 \subseteq X_2 \subseteq \cdots \subseteq X_n = X$. One can think of a filtration to be a specific way to grow and generate X. As we "filter" through the space X using this nested sequence of subsets, new topological features may be created and some older ones may be destroyed. Persistent homology tracks the creation ("birth") and destruction ("death") of these topological features with respect to the filtration index. For our purposes, we will consider a real valued index, which we will refer to as "time". The resulting births and deaths of features are summarized in a so-called *persistence* *diagram*. The persistence diagram is a set of points in the 2D plane whose (x, y) coordinates represent the birth and death times of the features. The life-time of a feature (death time—birth time) is called the *persistence* of this feature, encoding how long this feature exists during the filtration. Since its introduction, persistent homology has become a fundamental method to characterize/summarize shapes, as well as to separate significant features from "noise".

In our case, given a neuron tree T, we first choose one or more real-valued *descriptor functions* defined on T. These descriptor functions may encode purely geometric information, such as geodesic or Euclidean distance from a point on the tree, or functions encapsulating electrophysiological or dynamical information (such as the electrotonic distance from a base point). We then use topological persistence to summarize these descriptor functions through suitable filtrations of the neuron tree induced by the descriptor functions.

Descriptor functions. We will ignore the thickness of neuronal processes and represent the axonal or dendritic compartment of a neuron as geometric trees T embedded in 3D Euclidean space, consisting of tree nodes $V(T)$ and tree branches (curves connecting the tree nodes). We will use $|T|$ to denote the set of points belonging to the tree branches together with the tree nodes. A descriptor function is a real valued function $f : |T| \to \mathbb{R}$ defined on $|T|$. The thickness of neuronal processes can be encoded by appropriate descriptor functions.

The standard persistence summary that we introduce is defined on descriptor functions defined on a continuous domain. If the function values are specified only at tree nodes $V(T)$, we can extend these values to a *piecewise-linear (PL)* function $f : |T| \to \mathbb{R}$ on $|T|$ using linear interpolation along the length of the arc.

The main steps involved in the algorithm are shown in Fig 1. In the following, we use the following Euclidean distance descriptor function $f : |T| \to \mathbb{R}$ as an example:

Let r denote the root of T, which may be generically located in the soma of the neuron. The *Euclidean distance function* $f : T \to \mathbb{R}$ is defined such that, for any $x \in T$, $f(x)$ equals the negative of the Euclidean distance between x and the root r of T; that is, $-f$ measures how far each point x in $|T|$ is from the the soma. (We set f to be the negation of Euclidean distance to the root so that the root has the highest function value, which eases the description of the persistence diagram below. In experiments, this is not necessary.) See Fig 2A for an example, where, for illustration purpose, we ignore the geometric embedding of the neuron tree T and plot it so that the height of each point x equals $f(x)$. Hence we sometimes also refer to f as the "height" of a point.

Persistence diagram w.r.t. *f*. We now describe the persistence diagram summary induced by the so-called *sublevel set filtration* of this function f. In particular, let

$$T_t \quad = \{x \in |T| \mid x < f(t)\} \tag{1}$$

be the *sub-level set* of T w.r.t f at t. We will track the persistent features for the sequence of subspaces T_t's as t increases:

$$T_{t_0} \subseteq T_{t_1} \subseteq \cdots T_{t_n} = T, \quad \text{with} \quad t_0 \le t_1 \le \cdots t_n. \tag{2}$$

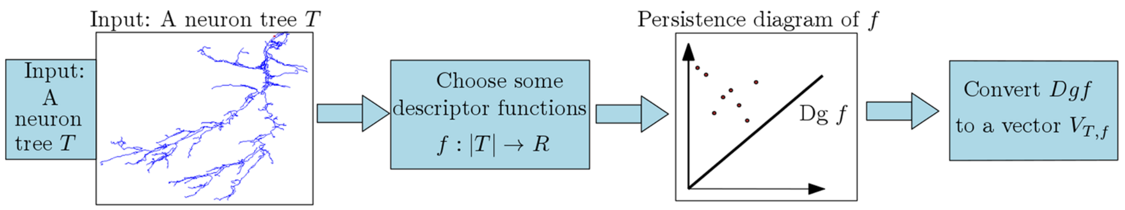

Input: A neuron tree T
Input: A neuron tree T
Choose some descriptor functions $f : |T| \to R$
Persistence diagram of f
Dg f
Convert Dgf to a vector $V_{T,f}$

Fig 1. Pipeline of persistence-based feature vectorization framework.

https://doi.org/10.1371/journal.pone.0182184.g001

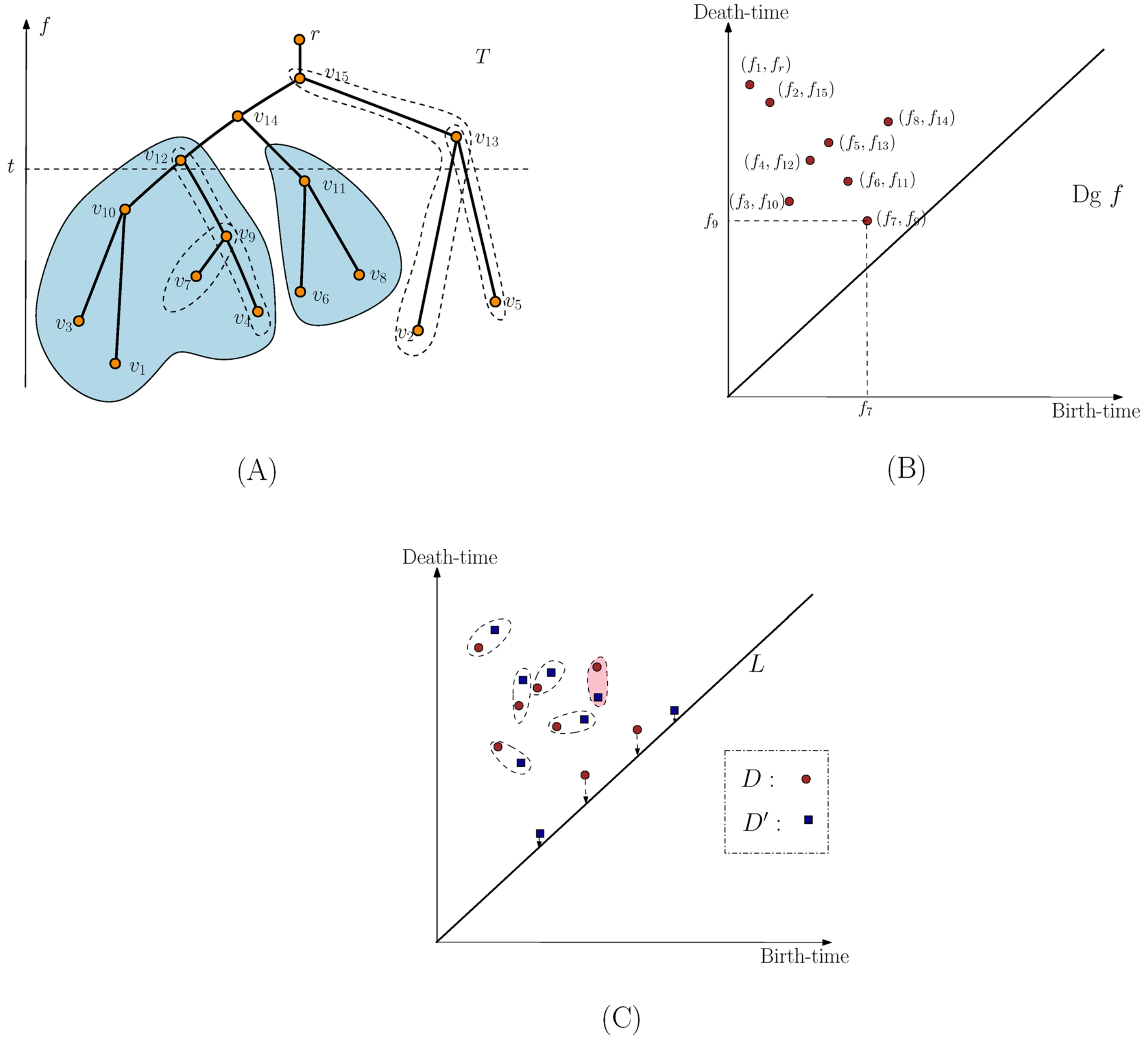


Fig 2. Examples of persistence diagrams. (A). We plot the tree T so that the height of a point is its f value. The sublevel set T_t is the portion of T lying below the horizontal dashed line corresponding to $\{x \in |T| \mid f(x) = t\}$. Consider what happens when the filtration index α passes the vertex v_{14}. At this time the left and right subtrees (shaded) merge at v_{14}. These subtrees were originally generated at $m_1 = v_1$ and $m_2 = v_6$ respectively. Since the right subtree was born at the later time $f(v_6) = f_6$, this event corresponds to the "death" of the right subtree (with a death time $f(v_{14}) = f_{14}$). This gives rise to a persistence point $(f(v_6), f(v_{14})) = (f_6, f_{14})$ in the persistence diagram in (B). In (B), for simplicity, we set $f_i := f(v_i)$. We mark some pairs of tree nodes generating persistence points in (A) via dashed closed curves, such as (v_7, v_9) and (v_4, v_{12}). In (C), red and blue points correspond to persistence points in two persistence diagrams D and D', respectively. An example of a correspondence is given, with points matched to diagonal considered to be noisy points.

https://doi.org/10.1371/journal.pone.0182184.g002

Since in our case T is simply a tree, T_t will consist of a set of disjoint pieces of the tree T, and birth/death events when a new disjoint piece appears (birth), or two disjointed pieces are joined (a death event for the shorter-lived of the two pieces involved). (Note that given the simple topology of the domain T, which is a tree, this provides only a simplified view of the general notion of persistent homology.)

More precisely, consider what happens as we sweep the tree with increasing height values. For any height t, we track the connected components in the portion of T with height smaller than t, which is exactly the sub-level set T_t: $= \{x \in |T| \mid f(x) \leq t\}$. As we sweep past a leaf node, a new component is *created* in the sub-level set. At a saddle point (a branching node), two or more components will be merged into a single one, and thus some components are *destroyed*. Note that each component (a subtree) in the sub-level set T_t is generated (created) by the global minimum in this component (intuitively, this is the first time any point in this component is created). Assume we sweep past a branching point s that merges two components, call them C_1 and C_2, into a single component C. Suppose C_1 and C_2 are generated by leaf nodes (minima) m_1 and m_2 respectively; and assume without loss of generality that $f(m_1) < f(m_2)$. Then intuitively after the merging, the "newer" component C_2 is destroyed and the component C_1 (created earlier at a smaller height) survives with m_1 generating the merged component C. As a result, we add a *persistent point* $(f(m_2), f(s))$ into the persistence diagram Dgf, indicating that a feature (branch) originally initiated at height $f(m_2)$ is killed at $f(s)$. The value $|f(s) - f(m_2)|$ is called the *persistence* of this branching feature, specifying its life-time. An example is shown in Fig 2. Two shaded subtrees merge at node v_{14}, which eliminates the subtree generated at v_6, giving rise to the persistent point (f_6, f_{14}) in the persistence diagram. Sweeping through the entire tree, we obtain a set of persistent points constituting the persistence diagram Dgf, each recording birth and death of branches in a *hierarchical manner* as induced by the distance function f. In this paper we use the *extended* persistence diagram, which includes the point (f_1, f_r) in the example shown, corresponding to inclusion of the maximum value of the distance function.

Intuitively, we can think of this procedure as a way to decompose the tree into a set of nested branching features (e.g. the feature (v_7, v_9) and (v_4, v_{12}) in Fig 2), each represented by a point $(b, d) \in Dgf$, recording its birth and death. Points with larger difference between coordinates have more persistence and so represent more robust features.

Super-level sets filtrations. In the above description, we swept the tree bottom up and inspected the changes in components of the sub-level set T_α during the sweep. This procedure captures the "merging" of branching features. Depending on the descriptor function, a general tree could have both down-fork and up-fork nodes (see Fig 3 where we assume that the height represents the function value of tree nodes). Symmetrically, we can also sweep the tree top-down and track the merging in components of the super-level set

$$T^t = \{x \in |T| \mid f(x) \geq t\} \tag{3}$$

as t decreases. This approach would give rise to a set of points recording the splitting-type branching features connected to up-fork tree nodes. We merge the two set of persistence diagrams into a single diagram $\widehat{\mathrm{Dg}f}$ (a single set of planar points), and call it the persistence summary induced by the descriptor function f.

Finally, the persistence summary can be efficiently computed in $O(n \log n)$ time for an input tree with n nodes. Note that this time can be improved to $O(n)$ time if one assumes that the descriptor function f is *monotonically increasing* along every tree path from root to a tree leaf; see the algorithm used by [30]. However, many natural descriptor functions (such as the

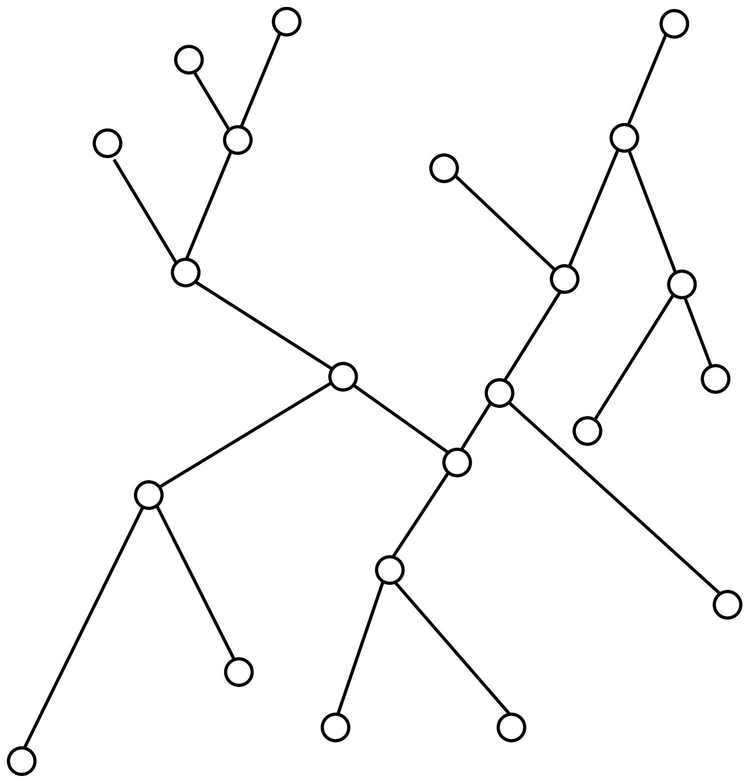


Fig 3. A general tree. For illustration purpose, height of a node represents its function value. A general tree may have both downfork and upfork branching nodes.

https://doi.org/10.1371/journal.pone.0182184.g003

Euclidean distance function) do not have this monotone property, and the $O(n)$ time complexity does not apply to those more general descriptor functions.

Step 2: Vectorization of persistence diagram summaries

Given a neuron tree T, we first construct a descriptor function f on it, and compute its persistence diagram summary $\widehat{\mathrm{Dg}}f$ induced by f. Given multiple neuron trees $T_1,\ldots, T_n$, we convert each of them to a persistence diagram summary $D_1, D_2,\ldots, D_n$. We now need an efficient way to compute distance between two persistence diagrams so as to compare the corresponding neurons. As we discuss in S1 File, the standard distance between persistence diagrams used in the topological data analysis literature is the so-called *bottleneck distance* (or its Wasserstein variant [25]). Intuitively, it identifies optimal "almost one-to-one" correspondence between points from one diagram to the other diagram, so that the maximum distance between pairs of corresponding points is minimized; and this minimal distance is the bottleneck distance between input persistence diagrams. (See Fig 2C for an illustration of a correspondence—some points are allowed to match to the diagonal $\mathrm{L} := \{(x, x) \mid x \in \mathbb{R}\}$, in which case they are considered noise.) While this is a natural way to measure distance between two persistence diagram summaries, its computation takes $O(k^{1.5} \log k)$ where k is the total number of persistent points in the diagram. Furthermore, this distance measure does not lend itself easily to fast searching and indexing. Therefore, in Step 2, we further vectorize the persistence diagram summaries, to map each persistence diagram into a point in $\mathbb{R}^d$ (i.e, a d-dimensional vector) as follows.

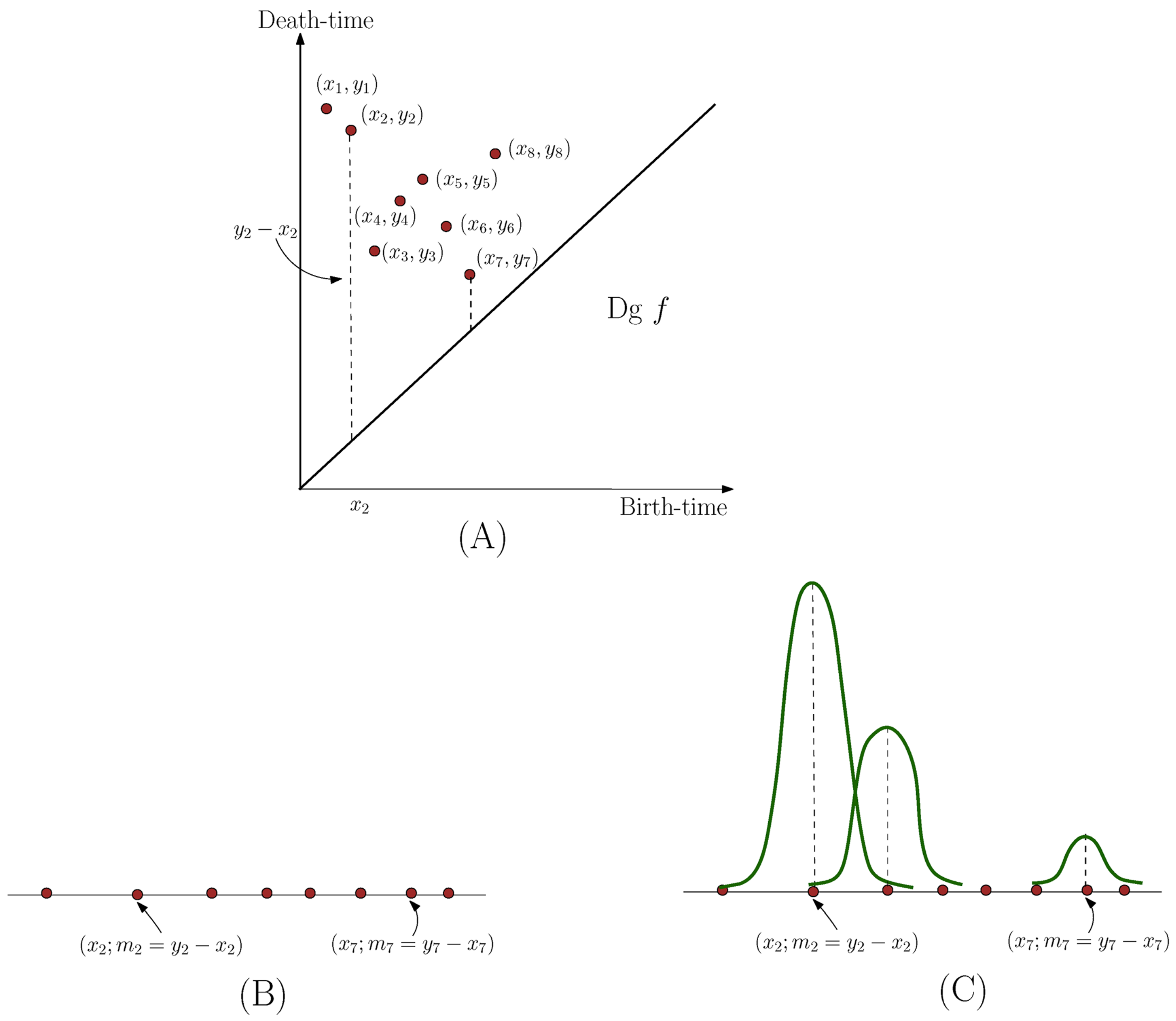


Fig 4. Converting persistence diagram to a 1D density function. We convert persistent points in the persistent diagram *Dgf* in (A) into a set of weighted points in the line as shown in (B). We then put a Gaussian function $m_i \cdot K_t(x_i, \cdot)$ at each point $x_i \in \mathbb{R}$, and the sum of them gives the function ρ_D. Note that a point with lower persistence (such as $(x_7, y_7 - x_7)$) has less contribution to the final density function ρ_D.

https://doi.org/10.1371/journal.pone.0182184.g004

Let D be a persistence diagram containing points $p_1, \ldots, p_k \in \mathbb{R}^2$. Recall that for each point $p_i = (x_i, y_i)$, its persistence is $|y_i - x_i|$, which is the vertical distance from p_i to the diagonal $L = \{(x, x) \mid x \in \mathbb{R}\}$. We can map p_i to a weighted point $\bar{p}_i \in \mathbb{R}$ at location x_i with mass $|y_i - x_i|$, which we represent as $\bar{p}_i = (x_i; m_i := |y_i - x_i|)$. See Fig 4 for an example. Next, we convert the collection of weighted 1D points $\{\bar{p}_i, i \in [1, k]\}$ into a 1D density using a simple kernel estimate:

$$\rho_D(x) := \sum_{i=1 \in k} m_i \cdot K_t(x, x_i), \text{ for any } x \in \mathbb{R}, \tag{4}$$

where $K_t(x, y) = e^{-\frac{(x-y)^2}{2t^2}}$ is a Gaussian kernel with width (standard deviation) t. We have a Gaussian function $g_i(x) = m_i\, K_t(x_i, x)$ centered around each x_i and the density function $\rho_D(x) = \Sigma_i\, g_i(x)$ is the sum of these Gaussian functions.

Recall that for a point $p_i = (x_i, y_i)$ in the persistence diagram, the persistence time $|y_i - x_i|$ measures its importance (how long it lives from its birth to death). The weighting of the Gaussian kernel by $m_i = |y_i - x_i|$ thus gives important features (with larger persistence) greater weights.

Finally, assume that the ranges of the birth / death times of all persistence points in D are contained in $[\mathtt{a}, \mathtt{b}]$ with $I = \mathtt{b} - \mathtt{a}$. We vectorize the density function ρ_D by a m-dimensional vector consisting of the function values at the m positions evenly spaced in the interval $[\mathtt{a}, \mathtt{b}]$.

$$v_D := \left[\rho_D\left(\mathtt{a} + \frac{I}{m}\right),\ \rho_D\left(\mathtt{a} + \frac{2I}{m}\right),\ \cdots\ , \rho_D\left(\mathtt{a} + \frac{mI}{m}\right) = \rho_D(\mathtt{b})\right]. \tag{5}$$

We call the above vector the *persistence-vector*. In our algorithm, we use the same range $[\mathtt{a}, \mathtt{b}]$ and m for all neuron structures, so that their resulting persistent-vectors are comparable.

The distance between two input neurons T_1 and T_2 can then be defined as the standard L_p-norm between the resulting vectors v_{D_1} and v_{D_2} obtained from their persistence profiles D_1 and D_2, respectively. That is, $d_V(T_1, T_2) := \|v_{D_1} - v_{D_2}\|_p$.

We remark that there has been several persistence-based profiles developed in the literature of topological data analysis, starting with the *persistence landscape* of [33]. We refer the readers to Section 2 of [34] for a summary of related work. Here we only mention two of the most relevant ones, the multi-scale descriptor of [35] and the persistent images of [34]. Unlike most other persistence-based profiles, both of these two approaches offer some stability guarantees. Our feature vectorization can be considered as a 1D version of the persistent images approach of [34]. We discuss the stability of persistence diagrams and our persistence feature vectors in S1 File.

Multiple descriptor functions

One advantage of our persistence feature vectorization framework is the generality of the descriptor function f. For example, we can use descriptor functions encoding morphometric measurements. Many quantities used in L-Measure can induce a descriptor function, such as: (i) define $f(v)$ to be the branch-angle spanned by the two child-branches of a tree node $v \in V(T)$; and (ii) define $f(v)$ to be the section area or the section radius of the branch at node v. We can also consider the *geodesic distance function* $g : |T| \to \mathbb{R}$, where $g(x)$ is defined as the geodesic distance to the root of the neuron tree T. The descriptor functions can also encode electrophysiological properties. Two such functions are voltage attenuation and propagation delay relative to a base point (e.g. soma), cf. the "morphoelectrotonic transform" [36].

Furthermore, we can encode more information about an input neuron by using multiple descriptor functions f_is, summarized in the map $F : |T| \to \mathbb{R}^r$:

$$F \quad = \langle f_1, \ldots, f_r \rangle : |T| \to \mathbb{R}^r;\ \text{that is, for each } i \in [1, r], f_i : |T| \to \mathbb{R}. \tag{6}$$

Given $F = \langle f_1, \ldots, f_r \rangle$ defined on T, we compute the persistence diagram $D_1, \ldots, D_r$ for each descriptor function. To aggregate these diagram into a single persistent vector, we use the following strategy:

We simply convert each D_i into a feature vector v_i, and concatenate them into a vector v_T of length rm.

If the dimension *rm* is too large, later, given a collection of neurons, we perform PCA to reduce the dimension of the resulting persistent-vectors to a lower dimension vector.

Potential extensions. As a future work, we will extend the persistence vectorization framework to characterize developmental dynamics of neuronal trees. In particular, the developmental process involves biologically important dynamic changes in neuronal trees. To reflect such changes, the persistence diagram can be extended to handle time-varying data. As a neuron's structure evolves, the corresponding persistence diagram varies, where each persistence point in it (a branching feature) traces out a curve, called a vine [37]. The evolution of all persistence points traces out a collection of vines, called a vineyard (Fig 5), which summarizes the evolution of a neuron's structure. Vines can terminate, or new vines can be created, corresponding to the disappearance of an existing branch or creation of a new one. The vectorization procedure can be extended to vineyards, possibly with an intermediate dimensionality-reduction step. Distributed activity measurements (trans-membrane voltage or local calcium concentration [38]) generate a time-varying function that can also be treated in this manner.

Connection to Sholl analysis

The persistent-vector for a given descriptor function provides more information than simple statistical summaries such as min, max or average values. Furthermore, by using a geometric descriptor function (such as Euclidean distance function and geodesic distance function), the persistence diagram can encode both local and global shape of the neuron trees, which has been challenging for most previous approaches in comparing neuron trees.

To illustrate the richness of information encoded in persistent summaries, below we show a connection between the persistent diagram *Dgf* of the Euclidean distance function *f* and the previously familiar *Sholl* analysis. Specifically, we show that one can recover quantities used in Sholl analysis from *Dgf*.

Recall that the Sholl analysis is based on the sequence of numbers $N(r)$ of (dendrite) intersections between a neuron structure and the concentric circle of increasing radius $r \in \mathbb{R}^+$, centered typically at the centroid of the cell body. One can treat this count N as a function $N : \mathbb{R}^+ \to \mathbb{R}^+$ w.r.to the radius $r \in \mathbb{R}^+$. Various Sholl-type approaches then performs further analysis, such as semi-log analysis of log-log analysis, to obtain one (or more) quantities to summarize this function. Hence the function N contains sufficient information for Sholl-type analysis. We call this function the *Sholl function N*.

Next, compute the persistence diagrams $Dg^{\perp} f$ and $Dg^{\top} f$ induced by the *sublevel set filtration* and the *super-level set filtration* induced by the Euculdean distance function *f*, respectively. Let $Dgf = Dg^{\perp} f \bigcup Dg^{\top} f$ be their union.

Now, consider the *level set* $f^{-1}(r)$: $= \{x \in |T| \mid f(x) = r\}$ of the function *f*. It can be seen that $N(r)$ is the number of connected components in the level set $f^{-1}(r)$. As we vary the radius r of the concentric circles, components in $f^{-1}(r)$ can appear, disappear, merge and split. The birth and death of components in the level-sets as r varies, are recorded by the persistence points in the two persistence-diagrams $Dg^{\perp} f \cup Dg^{\top} f$ (which is our summary *Dgf*). A persistent point $(b, d) \in Dgf$ indicates that a component is created in the level-set $f^{-1}(r)$ with $r = \min\{b, d\}$, either as a new component or the splitting of a previous component, and disappears or merges into another component in level-set $f^{-1}(r')$ with $r' = \max\{b, d\}$.

For a connected tree, the value $N(r)$, for any $r \in \mathbb{R}^+$, can be recovered by

$$N(r) \quad = |\{(b,d) \in \mathrm{Dg}^{\perp} \mid b \leq r, d \geq r\}| + |\{(b,d) \in \mathrm{Dg}^{\top} \mid b \geq r, d \leq r\}| - 1, \qquad (7)$$

where $|A|$ is the cardinality of a set A. See Fig 6 for an illustration, where $N(r)$ is the total

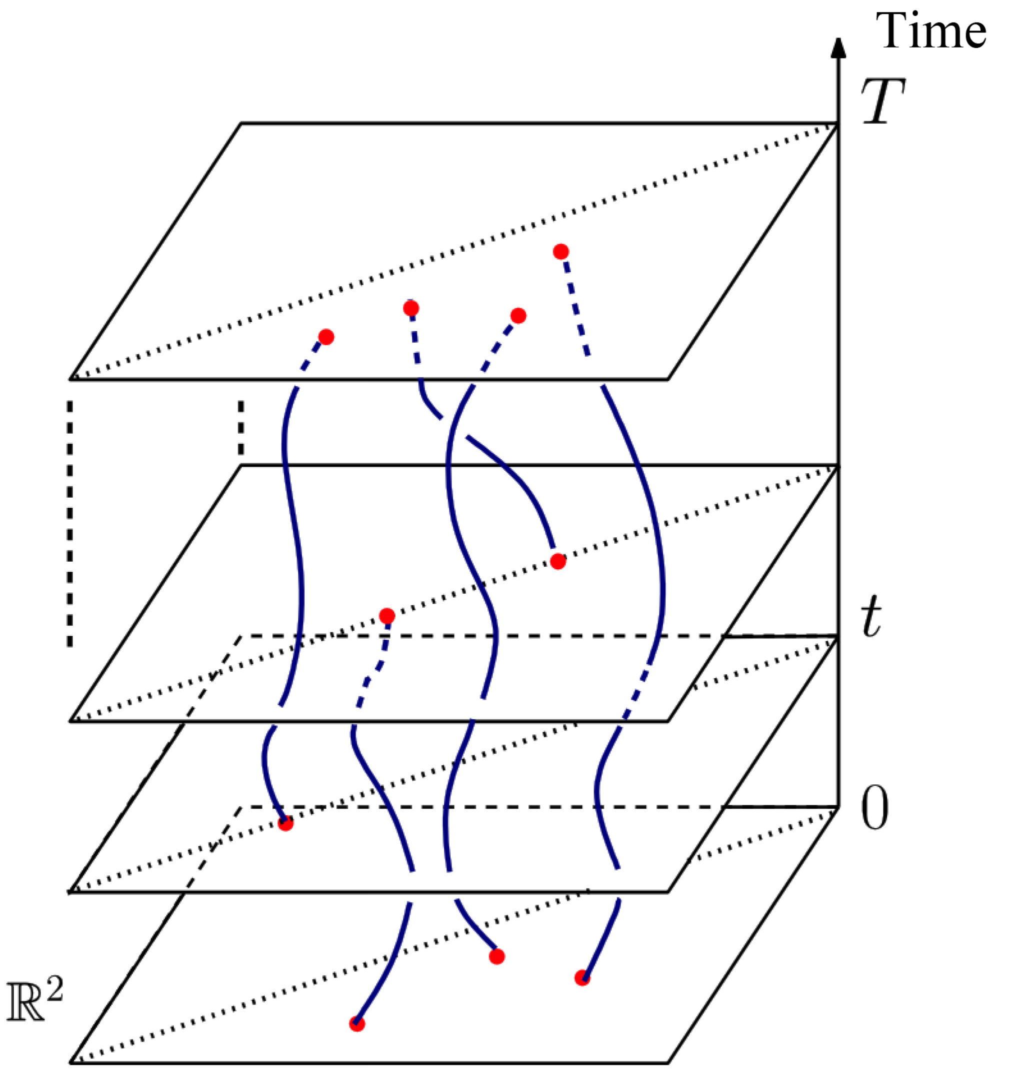


Fig 5. Vines and vineyard. Vertical direction specifies time, and each curve is a vine traced out by a persistent point as time varies.

https://doi.org/10.1371/journal.pone.0182184.g005

number of persistent points in the two shaded quadrants, reduced by one, to account for double counting of the soma or root node. Note that in this paper we utilize the extended persistence diagram (which includes the global maxima/minima of the height function).

In short, one can retrieve the Sholl function N for all r values from the persistent summary Dgf, and our persistence summary Dgf is strictly more informative than the Sholl function N. Specifically, while the Sholl function N records the number of components in the level set $f^{-1}(r)$, the persistent summary Dgf *tracks* these components—Indeed, as mentioned earlier and recall Fig 2, the persistent homology intuitively produces a hierarhical family of nested *branching features*, and each point in the persistence diagram encodes one such feature.

Nearest-neighbor classification accuracy based on feature vectors. Later in Section *Experimental results*, we will test the discriminative power of the persistence-based features. Given an input set of neurons $\mathcal{S}$, suppose we can compute some distance $d(T, T')$ between any two neurons $T, T' \in \mathcal{S}$ in it; such distance could be based on our persistence-based features, or

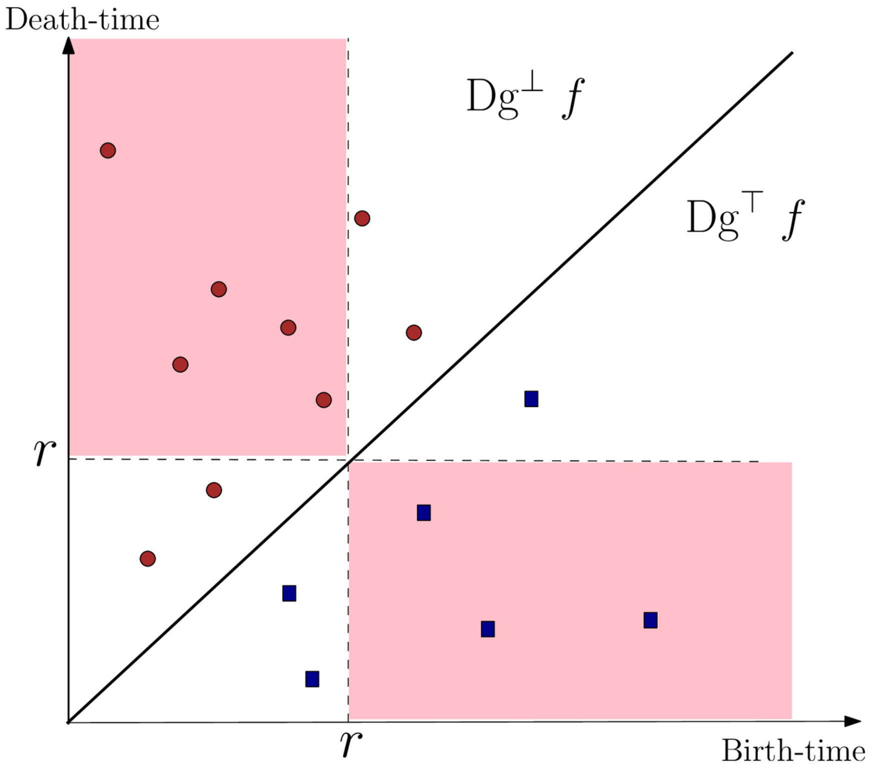


Fig 6. Illustration for relation between Sholl count and persistence diagram. Solid disks are points in $Dg^{\perp}$ *f* while squares are points in $Dg^{\top}$ *f*. *N*(*r*) equals to the number of points in the two shaded quadrants minus one (to correct for double counting of the root node). For the *r* value shown in the picture, *N*(*r*) = 7.

https://doi.org/10.1371/journal.pone.0182184.g006

other methods (say Sholl analysis) in our experiments. We will perform simple nearest-neighbor classification to decide the class membership of a query neuron: That is, given a query neuron T, we compute its nearest neighbor T^* in the training set under this distance d, and return the class membership of T^* as that for T. To test the accuracy of this simple classification, we perform the leave-one-out cross validation. Specifically, for each neuron $T \in \mathcal{T}$, we take $\mathcal{T} - \{T\}$ as the training set, and find its nearest neighbor in $\mathcal{T} - \{T\}$. We consider it a *success* if its nearest neighbor is from the same cell-type class as T.

We further extend this to the success-rate for the top k-nearest neighbor (k-NN): that is, given a neuron $T \in \mathcal{T} \backslash \{T\}$, we compute its k-nearest neighbors in $\mathcal{T} \backslash \{T\}$, and consider it a "success" (or a "*hit*") if these k-nearest neighbors include a neuron from the same family of T. The *success-rate* w.r.t. k-NN is defined as $SR_k = \frac{\text{hits}}{\text{neurons}}$.

Experimental results

In this section, we provide some preliminary experimental results as a proof-of-principle demonstration of our framework. In particular, we show the effectiveness of our method with just a single descriptor function. The input neuron is represented in the standard swc format. To simplify the tree, we assume that between any two tree nodes (which are nodes whose degree is not 2) there is a straight segment as arc. The weight of this arc is its Euclidean length (in other words, we ignore all the degree-2 nodes, and use the Euclidean distance between its two end points as the weight for an arc). We then use the *geodesic distance to the tree root r* as our test descriptor function; that is, $f : T \to \mathbb{R}$ where $f(x)$ is the total length of the unique tree path from the root r to point x. We choose this function as it captures the intrinsic metric structure of a neuron tree. As we report in (S2 File), using Euclidean distance function as the descriptor function gives typically worse performance in terms of the kNN-classification accuracy below; e.g, for `Dataset 1` and $k = 1$, using persistence-vectors (i.e, based on distance d_V), the success-rate is 0.4798 using Euclidean distance function, versus 0.5867 using our geodesic distance function. In all the experiments below, to convert a persistence diagram summary to a persistence-vector, we use a Gaussian kernel of width 50 (i.e, $t = 50$ in the kernel $K_t(x, y) = e^{-\frac{(x-y)^2}{2t^2}}$), and each feature vector is of dimension $m = 100$; see Eqs (4) and (5).

We use three test data sets below. `Dataset 1` [39] consists of 379 neurons, taken from the Chklovskii archive (Drosophila) of NeuroMorpho.Org, manually categorized into 89 types [22]. All the skeletons including the type information can be downloaded from http://neuromorpho.org under the 'Drosophila—Chklovskii' category.

`Dataset 2` contains 114 neurons from four families: Purkinje, olivocerebellar neurons, spinal motoneurons and hippocampal interneurons, downloaded also from NeuroMorpho.Org. Specifically, the 16 Purkinje reconstructions [40, 41, 42, 43] have only dendrites with no axons. The 68 olivocerebellar neurons reconstructions [44] have only axons, with no dendrites. The 17 spinal motoneurons reconstructions [45] have complete dendrites, but only the initial branches of the axons. In this case, we keep only the dendrites in our experiments. The 13 hippocampal interneurons reconstructions [46] have both dendrites and axons. In this case, we separate each hippocampal interneuron reconstruction into two trees: one for dendrites and one for axons. In total, we obtain 127 neuron trees, some of them dendritic and some axonal.

`Dataset 3` comes from the Human Brain Project [47] and are downloaded from NeuroMorpho.Org. It includes 1268 neuron cells, out of which the primary cell class (interneurons vs. principal cells) is known for 1130 cells. Both of these classes have complete dendrites. The interneurons have moderately complete axons, the principal cells have incomplete axons. We have not separated the dendrite and axonal trees in this case.

Nearest-neighbor classification accuracy

In this test, we aim to demonstrate the discriminative power of the persistence profiles. Given an input set of neurons $\mathcal{S}$, we compute the persistence diagram D_T for each of the input neuron $T \in \mathcal{S}$, and represent $\mathcal{S}$ by $\mathcal{D} = \{D_T \mid T \in \mathcal{S}\}$. We further vectorize these persistence diagrams and obtain a collection of feature vectors $\mathcal{V} = \{V_T \mid T \in \mathcal{S}\}$. To understand the effect of feature vectorization, below we will consider two distance metrics between neurons:

$$d_P(T_1, T_2) := d_{W,1}(D_{T_1}, D_{T_2}); \qquad d_V(T_1, T_2) := \|V_{T_1} - V_{T_2}\|_1. \tag{8}$$

That is, d_P is a distance based on the persistence diagram representation, defined as the degree-1 Wasserstein distance between the two persistence diagrams D_{T_1} and D_{T_2} (see Eq (2) in S1 File for the definition of 1-Wasserstein distance). d_V is the L_1-distance based on the persistence feature vector representation. For comparison purposes, we will also use a distance d_S based on Sholl-type analysis. In particular, given an embedded neuron structure T, we compute the Sholl function $N_T : \mathbb{R}^+ \to \mathbb{R}^+$ as introduced in Section *Connection to Sholl Analysis*, where $N_T(r)$ equals the number of intersections between the neuron tree T and the radius-r sphere centered at the root of tree T. Given two neurons T and T', intuitively, we would like to define the Sholl-based distance $d_S(T, T')$ as the L_1-norm of of $N_T - N_{T'}$. In our implementation, we discretize each Sholl function to a vector $\hat{N}_T$ of size 100 (which is the same as the size of discretization for the persistence feature vector), and compute the L_1-distance between the two vectors as $d_S(T, T') = \|\hat{N}_T - \hat{N}_{T'}\|_1$. We note that d_S directly compares the Sholl functions

Table 1. Leave-one-out cross validation for *k*-nearest neighbor classification rate where *k* = 1, 2, . . ., 5.

	# neurons classified correctly out of 346 neurons / success-rate, `Dataset 1`		
# nearest neighbors	Persist-distance d_P	Persist-vec d_V	Sholl-distance d_S
1	167 / 0.4827	203 / 0.5867	106 / 0.3064
2	193 / 0.5578	238 / 0.6879	146 / 0.4220
3	206 / 0.5954	258 / 0.7457	161 / 0.4653
4	220 / 0.6358	270 / 0.7803	173 / 0.5000
5	232 / 0.6705	273 / 0.7890	183 / 0.5289
	# neurons classified correctly out of 127 neurons / success-rate, `Dataset 2`		
# nearest neighbors	Persist-distance d_P	Persist-vec d_V	Sholl-distance d_S
1	116 / 0.9134	118 / 0.9291	79 / 0.3064
2	120 / 0.9449	121 / 0.9528	97 / 0.4220
3	121 / 0.9528	121 / 0.9528	103 / 0.4653
4	123 / 0.9685	123 / 0.9685	107 / 0.5000
5	123 / 0.9685	123 / 0.9685	110 / 0.5289
	# neurons classified correctly out of 1130 / success-rate, `Dataset 3`		
# nearest neighbors	Persist-distance d_P	Persist-vec d_V	Sholl-distance d_S
1	832 / 0.7363	794 / 0.7027	763 / 0.6752
2	992 / 0.8779	964 / 0.8531	942 / 0.8336
3	1055 / 0.9336	1030 / 0.9115	1019 / 0.9018
4	1094 / 0.9681	1064 / 0.9416	1058 / 0.9363
5	1111 / 0.9832	1088 / 0.9628	1081 / 0.9566

https://doi.org/10.1371/journal.pone.0182184.t001

(profiles) and thus tends to be more discriminative than using summary quantities, such as the area below the Sholl functions, or semi-log / log-log Sholl analysis, often used to compare neuron morphologies.

We then perform the simple k-nearest neighbor classification described at the end of Section *Material and methods*. The results are reported in Table 1, when in each entry we report both the number of hits and the success-rate (# hits / success-rate). For `Dataset 1`, we consider only those classes with at least 2 members, as, otherwise, it is meaningless to classify a neuron T when $\mathcal{T} \backslash \{T\}$ does not contain any neurons from the same family as T. This leaves 346 neurons. For `Dataset 3`, we consider only those neurons whose class-memberships are known, which gives us 1130 cells.

We observe that using d_V distance based on persistence vectors gives similar results as using the Wasserstein distance d_P between persistence diagrams—In the case of `Dataset 1`, the resulting classification accuracy based on distances d_V between persistence vectors are actually better than those based on persistence diagram distance d_P (e.g, 0.5867 versus 0.4827 for 1-nearest neighbor classification success rate). Using the Sholl function gives worst classification accuracy, especially when the number of nearest neighbors is small. The difference is particularly prominent for `Dataset 1`, which is much more diverse than other datasets (with around 80 classes) and more challenging to classify: E.g, the success-rate for 1-nearest neighbor classification is 0.5867 (using persistence vectors) versus 0.3064 (using Sholl vectors). To see the statistical robustness of these methods, we compute the mean and variance of the success-rate via bootstrapping—Results for `Dataset 1` are reported in Fig 7A, which are generated by taking 20 random subsamples of 250 neurons each.

We remark that the branch-density-based similarity measure proposed in [22] gives better classification accuracy over `Dataset 1`. However, we note that using the geodesic or

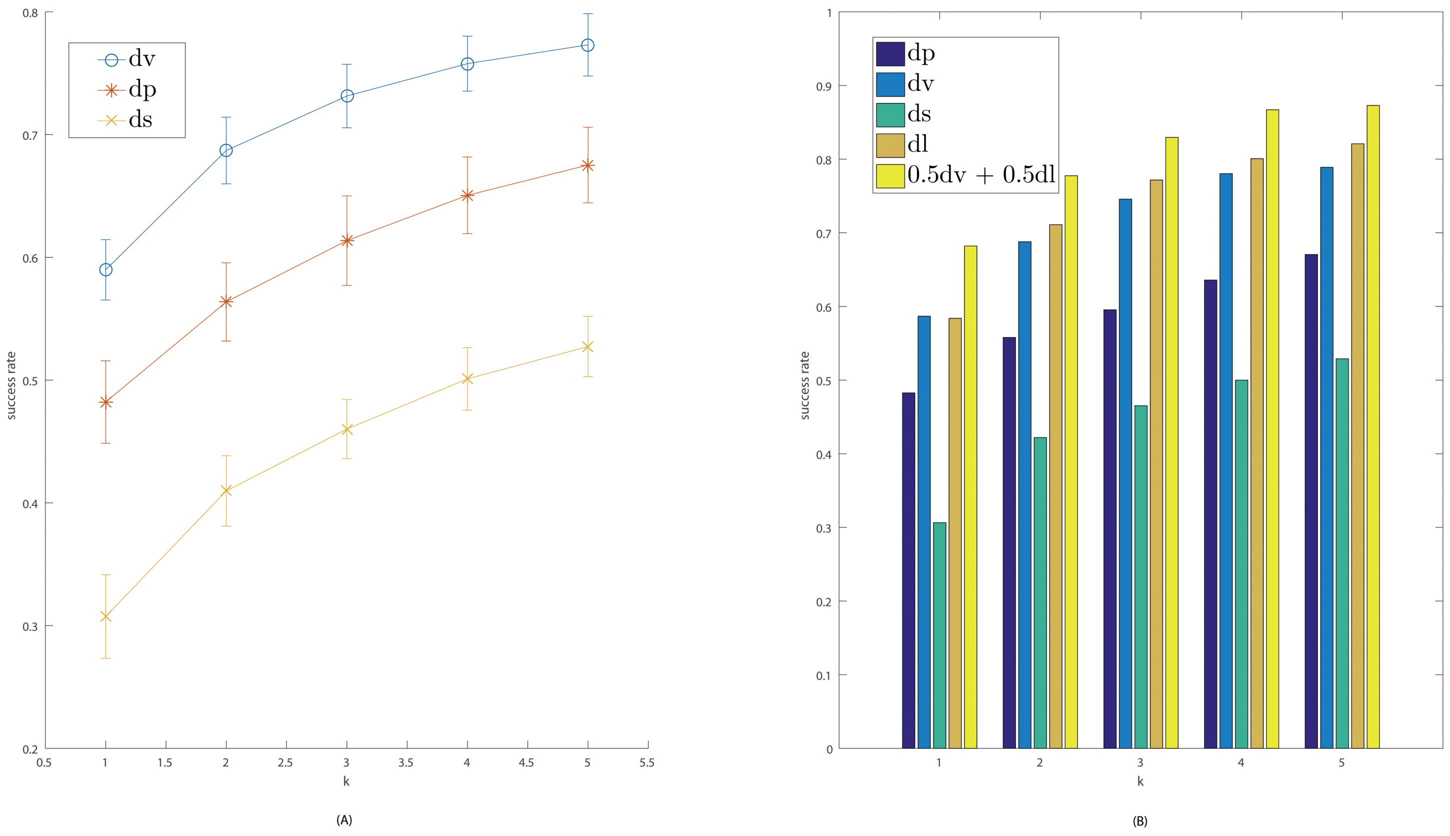


Fig 7. More results for Dataset 1. In both figures, *x*-axis represents value of *k*, and *y*-axis is the success-rate. (A) shows the mean and standard deviation (vertical bar) of the success-rates for *k*-nearest neighbor classification. (B) Comparison of success-rates based on persistence diagram distance d_P, persistence vector distance d_V, sholl-vector distance d_S, L-Measure based distance d_L, and a combined distance d_C (using persistence vector distance d_V and the L-Measure based distance d_L).

https://doi.org/10.1371/journal.pone.0182184.g007

Euclidean distance descriptor function, our approach is rigid-transformation invariant. The method of [22] however assumes that the input neurons are from the same coordinate system (with column / tangential directions given and already aligned).

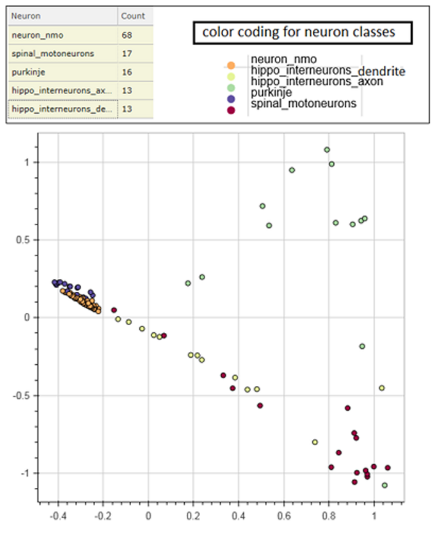
Comparison with L-Measure quantites

We also compare our persisentence-based features with specially-designed summaries of neuron morphology contained in L-Measure [4]. We use those L-measure parameters reported in the meta-data associated to each neuron in NeuroMorpho.Org; see S2 File for more details. If we use only a single individual measurement (say the "average bifurcation angle local"), the classification accuracy is very low—on average, the success rate is only 0.0919 based on individual measurements for $k = 1$ (see S2 File for the success-rate for each measurement).

We further combine all the measurements from NeuroMorpho.Org into a single feature vector, which we denote by L_T for a neuron T. Given a collection of neurons $\mathcal{S}$, we define the L-Measure based distance $d_L(T_1, T_2)$ between two neurons $T_1, T_2 \in \mathcal{S}$ by the normalized-L_2 distance between the corresponding vectors L_{T_1} and L_{T_2} (see S2 File for details). The comparison of success-rates of k-nearest neighbor classification based on different distances is given in Fig 7B. We observe that the L-Measure based distance gives very similar results to the persistence-based feature vectors (sometimes, L-Measure based distance could be slightly better, e.g, the success-rate is 0.7110 using d_L versus 0.6879 using d_V for $k = 2$). However, first, it is important to note that these measurements are specifically designed to capture neuron-morphology, while our persistence-feature achieves comparable classification performance with only a **single** geometric descriptor function: the geodesic distance to the root. For example, the L-Measure quantities used include angle information, surface area / volume, partition asymmetry, average Rall's ratio, etc. Secondly, very interestingly, when we combine the distances based on persistence-feature vectors and L-Measure based distance by defining $d_C = 0.5d_V + 0.5d_L$, we obtain **even better** accuracy as shown in Fig 7B: For example, for $k = 1$, the success rate is 0.5867 based on persistence vectors (d_V), 0.5838 based on L-Measure distance d_L, but 0.6821 based on the combined distance d_C, which represents an 17% improvement over using either d_V or d_L alone. This suggests that the information encoded in our persistence-feature based vectors has the potential to complement existing features, a direction that we will explore in more detail in the future.

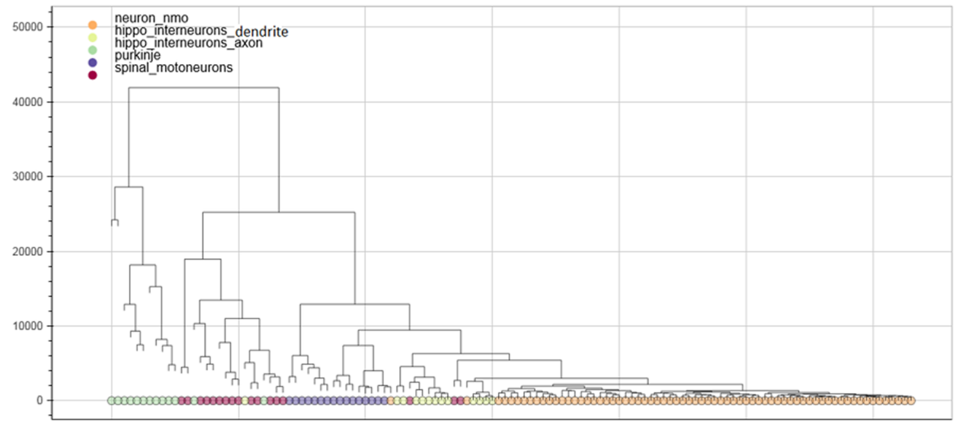
Clustering

We now explore the clustering structure of input neurons based on our persistence-based distance. In Fig 8A, we show the embedding of the 127 neurons in `Dataset 2` to the plane via Laplacian Eigenmap [48], which is a popular non-linear dimensionality reduction method. Each node in the plot represents a neuron, and its color reflects its neuron type. As we can see from the plot, neurons of different types are separated. To understand the clustering structure in more detail, we perform the so-called average linkage clustering method to produce a hierarchical clustering (HC) of the input neurons. In a hierarchical clustering tree (HCT), each leaf corresponds to an input neuron, each subtree represents a cluster, a down-fork node indicates the merging of two or more clusters (subtrees), and its height value corresponds to the distance threshold at which this merging happens. The HCT of `Dataset 2` is shown in Fig 8A, where leaf nodes (neurons) are marked by the same color coding as in Fig 8A.



Neuron	Count
neuron_nmo	68
spinal_motoneurons	17
purkinje	16
hippo_interneurons_ax...	13
hippo_interneurons_de...	13

(A)



(B)

Fig 8. Illustration for Dataset 2. (A) Embedding of `Dataset 2` in 2D via Laplacian Eigenmap. Each dot represents a neuron, and its color corresponds to its type. Its hierarchical clustering tree (HCT) is shown in (B), where each leaf corresponds to a neuron.

https://doi.org/10.1371/journal.pone.0182184.g008

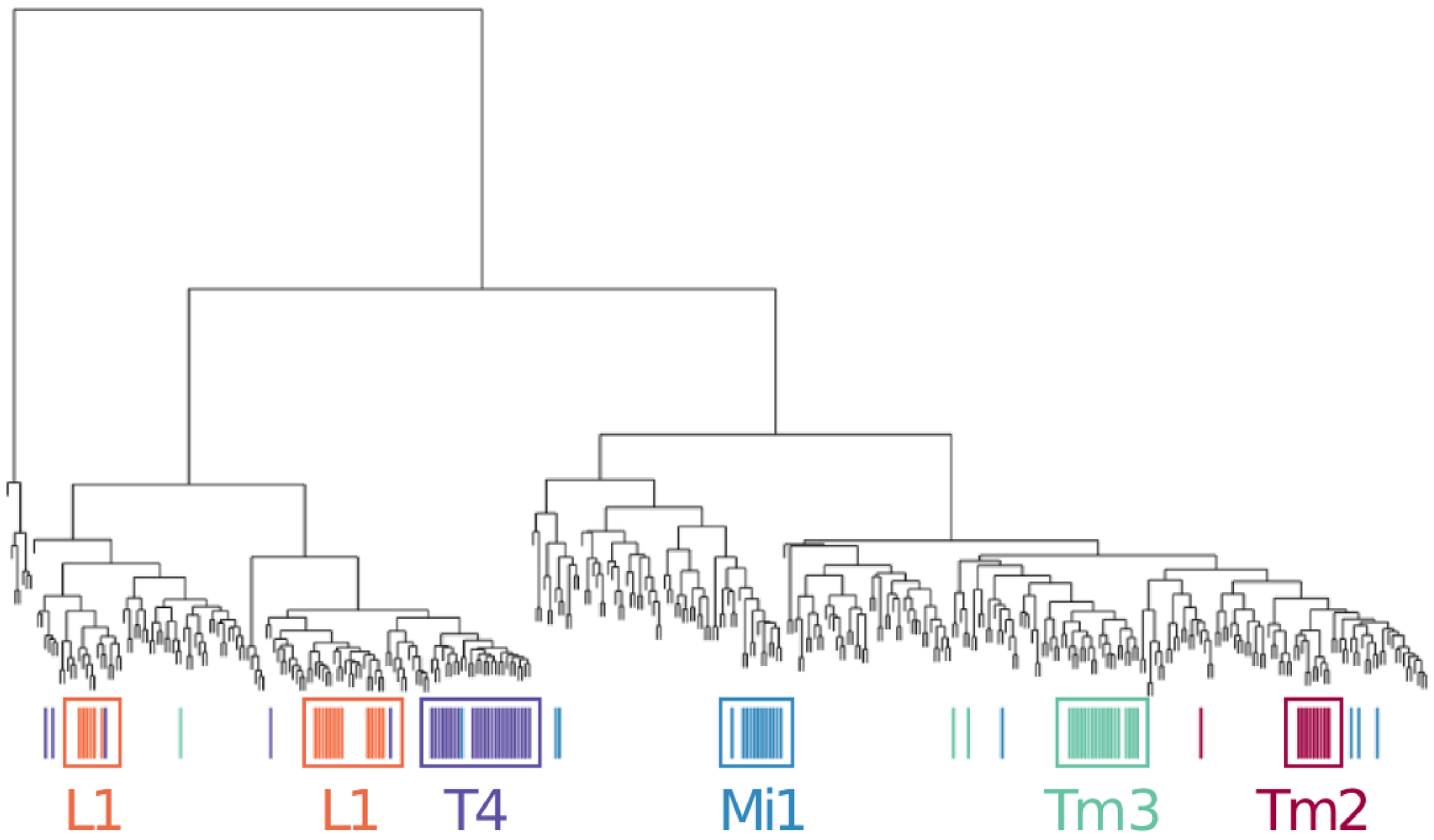


Fig 9. The HCT of `Dataset 1`. Each leaf corresponds to a neuron structure, and those from the top 5 largest classes (class "Tangential" excluded) are marked (color coded).

https://doi.org/10.1371/journal.pone.0182184.g009

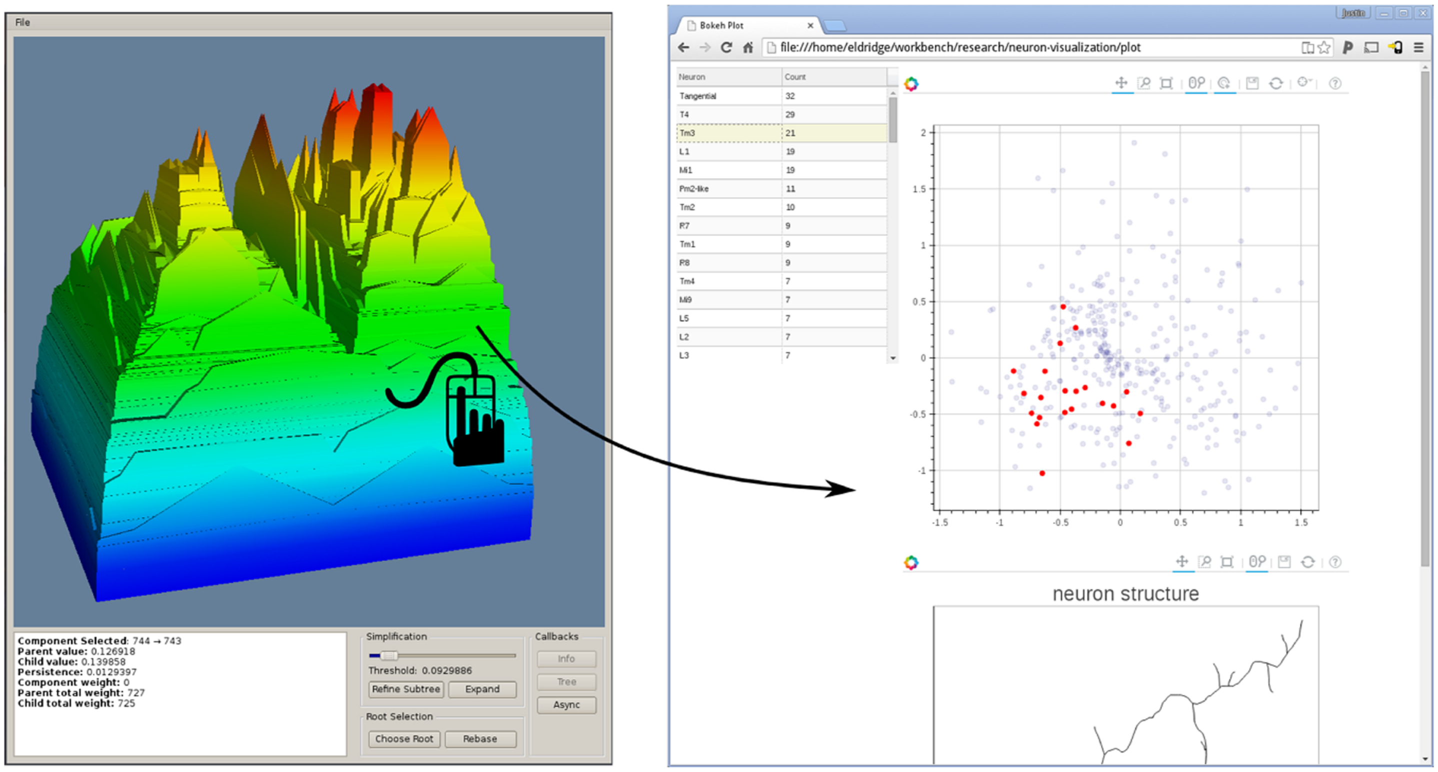
We choose a hierarchical clustering method since (1) it reveals more sub-clustering information than a flat clustering method, and (2) it permits the construction of a visualization platform later based on the hierarchical clustering structure to allow users to interactively explore the input data.

In Fig 9, we show the hierarchical clustering tree (HCT) for `Dataset 1`. Each leaf node corresponds to a neuron, and we mark those from the 5 largest of the 89 manually categorized classes [39] by colors—The largest class "Tangential" is excluded in this figure: Members from this class spread into several clusters in the HCT. This class is proved challenging to classify in the previous work as well [22]. As we can see, majority of neurons from each class are clustered together in the hierarchical clustering tree.

Visualization of the space of neurons

While the visualization of HCT in Figs 8 and 9 is useful in studying the clustering structure behind a collection of neurons, such a tree-visualization becomes ineffective for large data sets, mainly due to the cluttering of the large number of nodes. Indeed, the HCT already becomes hard to interact with for our `Dataset 3` with only a little more than 1000 structures. At the same time, the number of available neuron structures is rapidly increasing. For example, NeuroMorpho.Org holds about 50,000 structures just within a few years of its establishment.

Here we show a terrain visualization for the HCT, using an existing Denali software [49]. Specifically, instead of showing a tree, we build a terrain in 3D corresponding to the input



Component Selected: 744 → 743
Parent value: 0.126918
Child value: 0.139858
Persistence: 0.0129397
Component weight: 0
Parent total weight: 727
Child total weight: 725

Neuron	Count
Tangential	32
T4	29
Tm3	21
L1	19
Mi1	19
Pm2-like	11
Tm2	10
R7	9
Tm1	9
R8	9
Tm4	7
Mi9	7
L5	7
L2	7
L3	7

Fig 10. Terrain metaphor exploration tool for HCT. As a user selects a region in the terrain, the HCT view and embedding view are shown on the right. One can further select a neuron from the HCT / 2D embedding view, and inspects its structure as well as associated L-Measure information.

https://doi.org/10.1371/journal.pone.0182184.g010

HCT; see Fig 10. Each peak of the terrain corresponds to a cluster (i.e, the collection of nodes within some subtree in the HCT), and when two clusters merge in the HCT, their corresponding peak merges in the terrain. This terrain visualization platform provides many functionalities (see [49] for details), including allowing the coloring of the terrain based on a property of interest. A very important functionality is that the platform allows the user to explore a selected group of neurons in details, as well as to inspect each individual neuron structure. In particular, when a user clicks a specific region (corresponding to one cluster which main contain multiple levels of sub-clusters), our tool will return all the neurons contained in that cluster, and plot (i) the subtree rooted at this node, which corresponds to the multiple-levels of subclusters contained in this cluster and (ii) an embedding of all neurons in this cluster to the 2D plane via Laplacian Eigenmap. Note that these two types of visualization are not effective for large data sets, but effective now for a single cluster, which is typically of much smaller size. Furthermore, the user can select each individual neuron from either plots, and when a neuron is selected, its corresponding geometric structure will be shown in another panel which allows interactive manipulation. Other accompanying information (such as L-Measure values) will also be shown if available. This tool provides a way to explore the abstract space of neuronal morphologies using the persistence-based distance.

Discussion and conclusion

In this paper, we propose a generic framework to vectorize (linearize) the neuron structures and to compare them based on ideas from topological data analysis. Specifically, we use persistent homology to develop a meaningful summary of various types of information of neuron structures. For the example where the descriptor function encodes the Euclidean distance to the root (soma), we provide theoretical justification that its persistence summary is more informative than the standard Sholl-function.

As our proof-of-principle experiments demonstrate, even a single geodesic descriptor function can encode sufficient information about the morphology of neurons to provide coarse classifications for them. In particular, our persistence-based feature vectors provide significant better classification hit rates than Sholl-functions (see Table 1) for the most challenging data set (Dataset 1)—For example, our hit rate almost doubles that resulting from the Sholl-function based distance when $k = 1$. Dataset 1 contains around 80 families, and many families contain only a small number of neurons. In contrast, Dataset 2 and 3 each contains only very few families, and the structural difference between these families are relatively large. As a result, the Sholl-function based distance is also discriminative enough to differentiating neurons from different classes. Nevertheless, our persistence-based distance always outperform the Sholl-based distance, and even for the two easier datasets (Dataset 2 and 3), note that our hit rate is still noticably higher when the number of nearest-neighbors k is small (k = 1 or 2).

We have implemented the proposed persistence-based feature vectorization and comparison framework for neuronal structures. The open source code of our algorithms are available at https://github.com/Nevermore520/NeuronTools (see directory ./Experiments for all data used in this submission). There are several interesting directions for future work based on our tool:

1. First, it would be interesting to build a database of descriptor functions. As mentioned earlier, natural choices of the descriptor functions include: Euclidean distance function, geodesic distance function, various L-Measure based functions, voltage attenuation and propagation delay relative to the soma, and so on. Then, by testing with various choices of *single* descriptor function systematically, we can compare and identify which descriptor functions are more effective at differentiating different neuron structures.

2. It would be interesting to test whether different descriptor functions can complement each other to provide better (more discriminative) feature vectors when they are combined via the approach discussed in Section *Multiple descriptor functions*. As different descriptor functions may have different importance in differentiating neurons, it would be desirable to learn the relative weights for different descriptor functions using training data, to further improve the sensitivity of our persistence-based distance for neurons. The preliminary results based on the combined distance d_C as shown in Fig 7B shows the promise of this direction.

3. It would also be interesting to extend our persistence-based framework to analyze and characterize developmental dynamics of neuronal trees by modeling them as time-varying descriptor functions; see the discussion in Section *Multiple descriptor functions*.

Supporting information

S1 File. Stability of persistence-based signatures.
(PDF)

S2 File. Additional details for experimental results.
(PDF)

Acknowledgments

We thank Toufiq Parag for helpful discussions and Ting Zhao for help with using their neuron comparison algorithms. We sincerely thank Justin Eldridge for sharing his code and general help on generating HCT and terrain metaphor tool.

Author Contributions

Conceptualization: Giorgio A. Ascoli, Partha Mitra, Yusu Wang.

Data curation: Yanjie Li, Giorgio A. Ascoli.

Formal analysis: Yanjie Li, Giorgio A. Ascoli, Partha Mitra, Yusu Wang.

Funding acquisition: Giorgio A. Ascoli, Partha Mitra, Yusu Wang.

Investigation: Yanjie Li, Dingkang Wang, Giorgio A. Ascoli, Partha Mitra, Yusu Wang.

Methodology: Giorgio A. Ascoli, Partha Mitra, Yusu Wang.

Project administration: Partha Mitra, Yusu Wang.

Resources: Partha Mitra, Yusu Wang.

Software: Yanjie Li, Dingkang Wang.

Supervision: Partha Mitra, Yusu Wang.

Validation: Yanjie Li, Dingkang Wang, Giorgio A. Ascoli, Yusu Wang.

Visualization: Yanjie Li.

Writing – original draft: Yanjie Li, Partha Mitra, Yusu Wang.

Writing – review & editing: Yanjie Li, Giorgio A. Ascoli, Partha Mitra, Yusu Wang.

References

1. Ascoli GA, Dohohue DE, Halavi M. NeuroMorpho.Org: A central resource for neuronal morphologies. J Neurosci. 2007; 27(35):9247–51. https://doi.org/10.1523/JNEUROSCI.2055-07.2007 PMID: 17728438
2. Chiang AS, Lin CY, Chuang CC, Chang HM, Hsieh CH, Yeh CW, et al. Three-Dimensional Reconstruction of Brain-wide Wiring Networks in Drosophila at Single-Cell Resolution. Current Biology. 2011; 21(1):1–11. https://doi.org/10.1016/j.cub.2010.11.056 PMID: 21129968
3. Armañanzas R, Ascoli GA. Towards the automatic classification of neurons. Trends in Neurosciences. 2015; 38:307–318. https://doi.org/10.1016/j.tins.2015.02.004 PMID: 25765323
4. Scorcioni R, Polavaram S, Ascoli GA. L-Measure: A web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. Nature Protocols. 2008; 3(5):866–876. https://doi.org/10.1038/nprot.2008.51 PMID: 18451794
5. Lu Y, Carin L, Coifman R, Shain W, Roysam B. Quantitative Arbor Analytics: Unsupervised Harmonic Co-Clustering of Populations of Brain Cell Arbors Based on L-Measure. Neuroinformatics. 2015; 13(1): 47–63. https://doi.org/10.1007/s12021-014-9237-2 PMID: 25086878
6. Polavaram S, Gillette TA, Parekh R, Ascoli GA. Statistical analysis and data mining of digital reconstructions of dendritic morphologies. Frontiers in Neuroanatomy. 2014; 8(138). https://doi.org/10.3389/fnana.2014.00138 PMID: 25538569
7. Zawadzki K, Feenders C, Viana M, Kaiser M, Costa L. Morphological Homogeneity of Neurons: Searching for Outlier Neuronal Cells. Neuroinformatics. 2012; 10(4):379–389. https://doi.org/10.1007/s12021-012-9150-5 PMID: 22615032
8. Sholl DA. Dendritic organization in the neurons of the visual and motor cortices of the cat. Journal of Anatomy. 1953; 87:387–406.1. PMID: 13117757
9. Guerra L, McGarry LM, Robles V, Bielza C, Larranaga P, Yuste R. Comparison between supervised and unsupervised classifications of neuronal cell types: a case study. Dev Neurobiol. 2011; 71(1): 71–82. https://doi.org/10.1002/dneu.20809 PMID: 21154911

10. DeFelipe J, López-Cruz PL, Benavides-Piccione R, Bielza C, Larrañaga P, Anderson S, et al. New insights into the classification and nomenclature of cortical GABAergic interneurons. Nat Rev Neurosci. 2013; 14(3):202–16. https://doi.org/10.1038/nrn3444 PMID: 23385869

11. Santana R, McGarry LM, Bielza C, Larrañaga P, Yuste R. Classification of neocortical interneurons using affinity propagation. Frontiers in Neural Circuits. 2013; 7:185. https://doi.org/10.3389/fncir.2013.00185 PMID: 24348339

12. Sümbül U, Song S, McCulloch K, Becker M, Lin B, Sanes JR, et al. A genetic and computational approach to structurally classify neuronal types. Nature Communications. 2014; 5.

13. Costa M, Ostrovsky AD, Manton JD, Prohaska S, Jefferis GS. NBLAST: Rapid, sensitive comparison of neuronal structure and construction of neuron family databases. Neuron. 2016; 91(2):293–311. https://doi.org/10.1016/j.neuron.2016.06.012 PMID: 27373836

14. Gillette T, Grefenstette J. On Comparing Neuronal Morphologies with the Constrained Tree-edit-distance. Neuroinformatics. 2009; 7(3):191–194. https://doi.org/10.1007/s12021-009-9053-2 PMID: 19636974

15. Heumann H, Wittum G. The Tree-Edit-Distance, a Measure for Quantifying Neuronal Morphology. Neuroinformatics. 2009; 7(3):179–190. https://doi.org/10.1007/s12021-009-9051-4 PMID: 19475518

16. Zhang K, Jiang T. Some MAX SNP-hard results concerning unordered labeled trees. Inf Process Lett. 1994; 49:249–254. https://doi.org/10.1016/0020-0190(94)90062-0

17. Bille P. A survey on tree edit distance and related problems. Theor Comput Sci. 2005; 337(1-3): 217–239. https://doi.org/10.1016/j.tcs.2004.12.030

18. Wan Y, Long F, Qu L, Xiao H, Hawrylycz M, Myers EW, et al. BlastNeuron for Automated Comparison, Retrieval and Clustering of 3D Neuron Morphologies. Neuroinformatics. 2015; 13(4):487–499. https://doi.org/10.1007/s12021-015-9272-7 PMID: 26036213

19. Gillette TA, Brown KM, Ascoli GA. The DIADEM Metric: Comparing Multiple Reconstructions of the Same Neuron. Neuroinformatics. 2011; 9(2–3):233–245. https://doi.org/10.1007/s12021-011-9117-y PMID: 21519813

20. Basu S, Condron B, Acton ST. Path2Path: Hierarchical path-based analysis for neuron matching. In: Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on; 2011. p. 996–999.

21. Mottini A, Descombes X, Besse F. From Curves to Trees: A Tree-like Shapes Distance Using the Elastic Shape Analysis Framework. Neuroinformatics. 2015; 13(2):175–191. https://doi.org/10.1007/s12021-014-9255-0 PMID: 25391359

22. Zhao T, Plaza SM. Automatic Neuron Type Identification by Neurite Localization in the Drosophila Medulla. arXiv preprint. 2014;arXiv:1409.1892.

23. Gillette TA, Hosseini P, Ascoli GA. Topological characterization of neuronal arbor morphology via sequence representation: II—global alignment. BMC Bioinformatics. 2015; 16:209. https://doi.org/10.1186/s12859-015-0605-1 PMID: 26141505

24. Carlsson G. Topology and Data. Bull Amer Math Soc. 2009; 46:255–308. https://doi.org/10.1090/S0273-0979-09-01249-X

25. Edelsbrunner H, Harer J. Computational Topology—an Introduction. American Mathematical Society; 2009.

26. Lum PY, Singh G, Lehman A, Ishkanov T, Vejdemo-Johansson M, Alagappan M, et al. Extracting insights from the shape of complex data using topology. Scientific Reports. 2013; 3(1236). https://doi.org/10.1038/srep01236 PMID: 23393618

27. Edelsbrunner H, Letscher D, Zomorodian A. Topological persistence and simplification. Discrete Comput Geom. 2002; 28:511–533. https://doi.org/10.1007/s00454-002-2885-2

28. Zomorodian A, Carlsson G. Computing Persistent Homology. Discrete Comput Geom. 2005; 33(2): 249–274. https://doi.org/10.1007/s00454-004-1146-y

29. Edelsbrunner H, Morozov D. Persistent homology: theory and practice. In: European Congress of Mathematics. Europ. Math. Soc.; 2012. p. 31–50.

30. Kanari L, Dlotko P, Scolamiero M, Levi R, Shillcock J, Hess K, et al. Quantifying topological invariants of neuronal morphologies. arxiv print. 2016;abs/1603.08432.

31. Dey TK, Shi D, Wang Y. Comparing graphs via persistence distortion. In: Proc. 31rd Annu. Sympos. Comput. Geom. (SoCG); 2015. p. 491–506.

32. Skwerer S, Pieloch A, Miller E, Marron JS, Bendich P. Persistent homology analysis of brain artery trees. The Annals of Applied Statistics. 2016; 10(1):198–218. https://doi.org/10.1214/15-AOAS886 PMID: 27642379

33. Bubenik P. Statistical topological data analysis using persistence landscapes. Journal of Machine Learning Research. 2015; 16(1):77–102.

34. Adams H, Chepushtanova S, Emerson T, Hanson E, Kirby M, Motta F, et al. Persistent Images: A Stable Vector Representation of Persistent Homology; 2015.

35. Reininghaus R, Bauer U, Huber S, Kwitt R. A Stable Multi-scale Kernel for Topological Machine Learning. In: Proc. 2015 IEEE Conf. Comp. Vision & Pat. Rec. (CVPR); 2015. p. 4741–4748.

36. Zador AM, Agmon-Snir H, Segev I. The morphoelectrotonic transform: a graphical approach to dendritic function. J Neurosci. 1995; p. 1669–1682. PMID: 7891127

37. Cohen-Steiner D, Edelsbrunner H, Morozov D. Vines and Vineyards by Updating Persistence in Linear Time. In: ACM Sympos. Comput. Geom. (SoCG); 2006. p. 119–126.

38. Ginger M, Broser P, Frick A. Three-dimensional tracking and analysis of ion channel signals across dendritic arbors. Frontiers in Neural Circuits. 2013; 7:61. https://doi.org/10.3389/fncir.2013.00061 PMID: 23576958

39. Takemura SY, Bharioke A, Lu Z, Nern A, Vitaladevuni S, Rivlin PK, et al. A visual motion detection circuit suggested by Drosophila connectomics. Nature. 2013; 500(7461):175–181. https://doi.org/10.1038/nature12450 PMID: 23925240

40. Rapp M, Segev I, Yarom Y. Physiology, morphology and detailed passive models of guinea-pig cerebellar Purkinje cells. The Journal of Physiology. 1994; 474(1):101–118. https://doi.org/10.1113/jphysiol.1994.sp020006 PMID: 8014888

41. Vetter P, Roth A, Häusser M. Propagation of Action Potentials in Dendrites Depends on Dendritic Morphology. Journal of Neurophysiology. 2001; 85(2):926–937. PMID: 11160523

42. Chen XR, Heck N, Lohof AM, Rochefort C, Morel MP, Wehrlé R, et al. Mature Purkinje Cells Require the Retinoic Acid-Related Orphan Receptor-*α* (ROR*α*) to Maintain Climbing Fiber Mono-Innervation and Other Adult Characteristics. The Journal of Neuroscience. 2013; 33(22):9546–9562. https://doi.org/10.1523/JNEUROSCI.2977-12.2013 PMID: 23719821

43. Martone ME, Zhang S, Gupta A, Qian X, He H, Price DL, et al. The cell-centered database. Neuroinformatics. 2003; 1(4):379–395. https://doi.org/10.1385/NI:1:4:379 PMID: 15043222

44. Brown KM, Sugihara I, Shinoda Y, Ascoli GA. Digital morphometry of rat cerebellar climbing fibers reveals distinct branch and bouton types. The Journal of Neuroscience. 2012; 32(42):14670–14684. https://doi.org/10.1523/JNEUROSCI.2018-12.2012 PMID: 23077053

45. Li Y, Brewer D, Burke RE, Ascoli GA. Developmental changes in spinal motoneuron dendrites in neonatal mice. The Journal of Comparative Neurology. 2005; 483(3):304–317. https://doi.org/10.1002/cne.20438 PMID: 15682391

46. Ascoli GA, Brown KM, Calixto E, Card JP, Galván EJ, Perez-Rosello T, et al. Quantitative morphometry of electrophysiologically identified CA3b interneurons reveals robust local geometry and distinct cell classes. The Journal of Comparative Neurology. 2009; 515(6):677–695. https://doi.org/10.1002/cne.22082 PMID: 19496174

47. Markram H, Muller E, Ramaswamy S, Reimann MW, Abdellah M, Sanchez CA, et al. Reconstruction and Simulation of Neocortical Microcircuitry. Cell. 2015; 163:456–492. https://doi.org/10.1016/j.cell.2015.09.029 PMID: 26451489

48. Belkin M, Niyogi P. Laplacian Eigenmaps for dimensionality reduction and data representation. Neural Computation. 2003; 15(6):1373–1396. https://doi.org/10.1162/089976603321780317

49. Eldridge J, Belkin M, Wang Y. Denali, A visualization tool for tree like structures, URL: http://denali.cse.ohio-state.edu/;