Convolutional Persistence for Cubical Complexes

Elchanan Solomon
Department of Mathematics,
Duke University
Durham, USA
yitzchak.solomon@duke.edu

Paul Bendich
Department of Mathematics, Duke University
Geometric Data Analytics
Durham, USA
paul.bendich@duke.edu

***Abstract*—todo.**

I. Overview

Persistent homology is a method of assigning multiscale topological descriptors to parametric families of shapes. In *functional* persistence, the object of study is a real-valued function $f : X \to \mathbb{R}$ defined over a space X, and the parametric family of shapes are the sublevel-sets $X_\alpha = \{x \in X : f(x) \leq \alpha\}$. It is similarly possible to consider superlevel-sets, which is equivalent to negating the *filter function* f. One crucial feature of this construction is that X_α is a subset of X_β for $\alpha \leq \beta$, so that the sublevel-sets are naturally nested. The output of persistent homology is a collection of intervals (equivalently, a collection of points), called a *barcode* (or *persistence diagram*, using the point representation). The space of barcodes is not a vector space, even approximately **(cite!)**, but there do exist multiple *vectorizations* that transform barcodes into vectors suitable for machine learning and data analysis.

A very general setting for functional persistence is that of *cubical complexes*, i.e. shapes obtained by gluing together cubical regions like pixels or voxels. For example, if X is a 2D rectangular grid, a function $f : X \to [0, 1]$ can be viewed as a greyscale image. Persistent homology can then be understood as a feature extraction method for such images, either for supervised or unsupervised learning. Example applications include **cite: removing image noise (Chung), parameter estimation for PDEs (Calcina, Adams), segmentation (Chen), flow estimation (Suzuki), tumor analysis (Crawford), cell immune micronenvironment (Carriere), and materials science (Hiraoka)**.

It should be understood that purely topological methods do not provide state-of-the-art predictive accuracy on most machine learning tasks, and are not to be considered as *alternatives* to more general methods like neural nets and kernel methods. Rather, topological methods provide *principled* and *intepretable* features that are different from those extracted by other methods, and can help improve the performance and utility of the entire pipeline.

The authors were partially supported by the Air Force Office of Scientific Research under the grant "Geometry and Topology for Data Analysis and Fusion", AFOSR FA9550-18-1-0266.

To that end, it is important to understand the properties of functional persistence and their role in machine learning. Here we consider some of the most salient features:

- (Computational complexity) For a cubical complex K with N cubes, computing persistence is $O(N^\omega)$, where ω is the matrix multiplication constant. This means that such persistence calculations scale poorly in the resolution of data, especially high-dimensional data, where doubling the resolution in $\mathbb{R}^d$ results in a 2^d-fold increase in the number of simplices.
- (Stability) Persistent homology is stable to small perturbations of the input data, in that the distance between the barcodes for two functions f and g on a common space X is bounded by $\|f - g\|_\infty$. However, persistence is not stable to outliers, so images that look similar outside of a small fraction of pixels can produce wildly different barcodes.
- (Flexibility) There is a single persistence diagram to be associated with each pair (X, f) of space X and real-valued function f. This lack of additional parameters makes applying persistence straightforward, but can also be limiting in contexts where featurizations should be data-dependent.
- (Invertibility) There exist many space-function pairs (X, f) and (Y, g) producing identical barcodes. Thus, persistence is not invertible as feature map, and this loss of information may hinder the capacity of persistence-based methods to identify patterns or distinguish distinct images.

Our goal in this paper is to introduce a modification to persistence of cubical complexes that is (1) faster to compute, (2) more stable to outliers, (3) allows for data-driven tuning, and (4) is provably invertible. Essentially, this technique consists of passing multiple convolutional filters over an image and computing the persistence of the resulting collection of low-resolution outputs. Since this pipeline consists principally of combining convolutions with persistence, we call it *convolutional persistence*.

The remainder of the paper is organized as follows. Section II provides a thorough, non-technical survey on persistent homology. Section III reviews related work, and places *convolutional persistence* in conversation with prior work on inverse problems in computational topology. Section IV contains the major theoretical results of the paper, with the highlight being

Theorem ? proving injectivity for convolutional persistence. Section V compares ordinary and convolutional persistence on a host of datasets, showing the capacity of convolutional persistence to produce features well-suited for image classification. Finally, Section VI discusses outstanding questions and generalizations of this work.

II. Background

The content of this paper assumes familiarity with the concepts and tools of persistent homology. Interested readers can consult the articles of Carlsson [Car09] and Ghrist [Ghr08] and the textbooks of Edelsbrunner and Harer [EH10] and Oudot [Oud15]. We include the following primer for readers interested in a high-level, non-technical summary.

A. Persistent Homology

Persistent homology records the way topology evolves in a parametrized sequence of spaces. In the case of functional persistence, we consider an ambient space X filtered by a real-valued function $f : X \to \mathbb{R}$. When X is a cubical complex, made by gluing together cubes of varying dimension (vertices, edges, squares, cubes, etc.), this construction is particularly simple. To every cube σ we associate a real value $f(\sigma)$ that encodes the time at which it appears in the filtration of X. The only restriction on the function f is the following consistency condition: if σ is a sub-cube of a higher-dimensional cube τ (i.e. an edge which sits at the boundary of a square), we must have $f(\sigma) \leq f(\tau)$, ensuring that cubes do not appear before any of their faces.

Given a filtered cubical complex, as the sequence of sublevel-sets evolves, the addition of certain edges or higher-dimensional simplices alters the *topological type* of the space. A precise way of quantifying topology is *homology*, which measures the number of connected components (zero-dimensional homology), cycles (one-dimensional homology), or voids (higher-dimensional homology) in a space. Thus, homology can change when two connected components merge or a new cycle is formed. Simplices responsible for such topological changes are called *critical*. Persistent homology records the parameter values at which critical simplices appear, notes the dimension in which the homology changes, and pairs critical values by matching the critical value at which a new homological feature appears to the critical value at which it disappears. This information is then organized into a structure called a *barcode*, which is simply a collection of intervals. Figure II.1 shows the computation of the zero- and one-dimensional barcodes for a simple cubical complex.

When working with cubical complexes, it is possible to limit the maximal dimension of simplices allowed in the construction. Given an arbitrary cubical complex K, we write K^m to denote the subcomplex consisting of all cubes of dimension at most m; this is called the *m-skeleton of K*. Note that K and K^m have the same homology in dimensions less than m, but may differ in dimension m, and K^m has no homology in dimension greater than m.

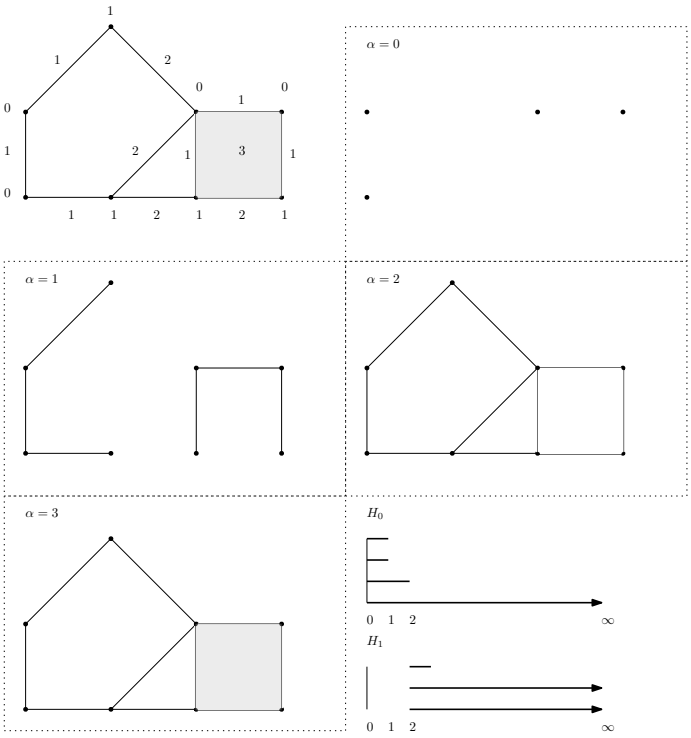

Fig. II.1: Top-left: A cubical complex with filtration values attached to vertices, edges, and squares. Top-right through bottom-left: sublevel-sets associated with different threshold values. Bottom-right: barcodes in dimensions zero and one. The zero-dimensional homology H_0 barcode contains four bars, since at $\alpha = 0$ there are four connected components. Two bars die at $\alpha = 1$, since at that threshold value there are only two connected components. Finally, $\alpha = 2$ sees the merger of these connected component, so another bar dies at $\alpha = 2$ and the last persists to infinity. In one-dimensional homology H_1, three bars are born at $\alpha = 2$, when three loops appear in the sublevel-set, and one of these bars dies at $\alpha = 3$, when that loop is killed off by the introduction of a square.

B. Image Cubical Complexes

Given a d-dimensional grayscale image, there are two ways of turning this data into a cubical complex. One is to view the voxels as being vertices, and higher-dimensional cubes as coming from voxels adjacencies, so that pairs of adjacent pixels form an edge and squares come from four voxels in a square formation, etc. There is a canonical way of extending the function f from the voxels (vertices) to the entire complex: given a cube τ, define $f(\tau)$ to be $\max_\sigma f(\sigma)$, where the max is taken over all vertices $\sigma < \tau$. Thus, a square appears precisely once all its constituent vertices appear; this is called the *lower-** filtration.

Alternatively, one can also view the voxels as being d-dimensional cubes, and have the lower-dimensional cubes

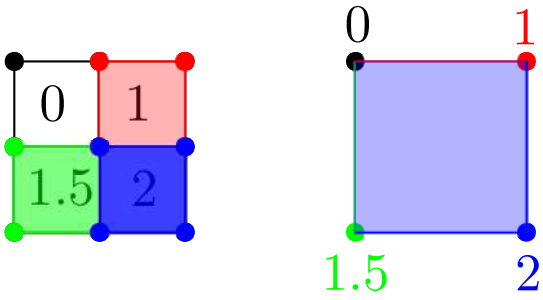

Fig. II.2: A 2×2 image can be turned into a complex in one of two ways. Left: A complex with four top-dimensional cubes, with function values, indicated using color, extended to vertices and edges via the lower-* rule. Right: A complex with four vertices, with function values extended to the edges and interior square via the lower-* rule.

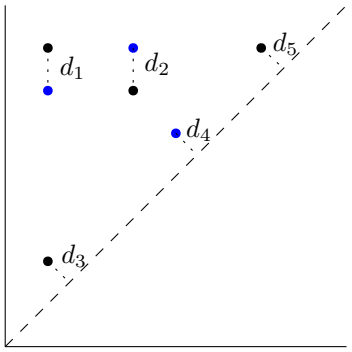

Fig. II.3: Two persistence diagram D_1 and D_2, one in black and the other in blue. An optimal matching between these diagrams is shown, containing both diagonal and non-diagonal pairings. The resulting p-Wasserstein distance is then $W_p(D_1, D_2) = \sqrt[p]{d_1^p + d_2^p + d_3^p + d_4^p + d_5^p}$.

be the faces of these vertices. As before, there is a canonical way of extending the function value from the voxels (top-dimensional cubes) down to entire complex: for a cube σ, define $f(\sigma)$ to be $\min_\tau f(\tau)$, where the min is taken over all voxels τ that contain σ. Thus, a cube appears precisely when at least one of the voxels in which it participates does; this is called the *upper-** filtration.

Consult figure II.2 for an illustration of these two images complexes. Generally, these complexes will differ, and the resulting persistence barcodes will be different. However, there exists a formula for reading the barcodes for one construction from the barcodes of the other, see [BGH$^+$21]. In this paper, we will adopt the former construction, taking our voxels to be the vertices of the cubical complex. **Is this what GIOTTO does?**

C. Comparing Persistence Diagrams

As multi-sets of points in the plane, persistence diagrams are not vectors. However, there exist multiple metrics for comparing persistence diagrams. The most common approach is to view persistence diagrams as discrete distributions on the plane $\mathbb{R}^2$, and use techniques from optimal transport theory, such as p-Wasserstein metrics W_p, to compare them. This analogy is complicated by the fact that persistence diagrams do not all have the same number of points, and that points in a persistence diagram near the diagonal line $y = x$ correspond to transient homological features, dying shortly after they are born, which ought not to play an important role in dictating similarity of diagrams. These problems are ameliorated by modifying the optimal transport protocol to allow paring points in one diagram either with points in the other diagram or with the diagonal line $y = x$, the latter incurring a cost proportional to the distance of the given point from the diagonal. This *diagonal paring* allows us to define an optimal transport distance between any pair of diagrams, regardless of how many points they have. In practice, the p-Wasserstein metrics in use are either $p = 1, p = 2$, or $p = \infty$, the latter of which is called the *Bottleneck distance* in persistence theory, and is written d_B. See Figure II.3 for a visualization of a matching between persistence diagrams.

D. Computational Complexity

Persistence calculations are $O(N^\omega)$, where N is the number of simplices in the filtration and ω is the matrix multiplication constant [MMS11]. For a metric space X, the number of $(d+1)$-dimensional simplices in the Rips complex is $\binom{|X|}{d+2}$, which are needed for computing persistence in degree d. Thus the computational complexity of d-dimensional Rips persistence is $O(\binom{|X|}{d+2}^\omega)$, which is huge even for small values of d. Though this bound is in practice quite pessimistic, it is accurate in reflecting the poor scaling of persistence calculations in $|X|$ [OPT$^+$17].

The bottleneck distance between persistence diagrams can be computed exactly using the well-known Hungarian algorithm, whose complexity is $O(n^3)$ for pairs of diagrams with at most n points. The number of points in the persistence diagram of filtration is always bounded by the total number N of simplices in the filtration, and may be much smaller. If one is interested in *approximating* the bottleneck, or more generally, p-Wasserstein distances between persistence diagrams, there are significantly faster methods than the Hungarian algorithm. One such method is based on *entropic approximation* techniques from optimal transport [LCO18], and provided that the norms of the points in the diagrams are uniformly bounded by some constant C, a transport plan within ϵ of the optimal matching can be produced in $O(\frac{n \log(n) C}{\epsilon^2})$ iterations of the Sinkhorn algorithm, which is itself linear in n. This algorithm is implemented in the persistent homology library GUDHI that we use in our experiments, and hence the computational complexity of our framework as a whole is dominated by the calculation of the diagrams themselves, and not the p-Wasserstein distances between them.

E. Properties of Persistent Homology

Persistence theory guarantees that a small modification to the filtration of a metric space produces only small changes in its persistence diagram. To be precise, if the appearance time of any given simplex is not delayed or advanced by more than δ, the persistence diagram as a whole is not moved by more than δ in the Bottleneck distance [CSEH07]. This implies that a small error in distance measurements in X produces only small distortion in the resulting persistence diagram. However, persistent homology is not at all stable to *outliers*, i.e. a small subset of the data having large error [BCD$^+$14].

Another important feature of persistent homology is that the pipeline mapping a metric space to a persistence diagram is *almost everywhere differentiable* [GHO16, PSO18]. This is because the coordinates of the points in the diagram correspond locally to fixed critical simplices in the data. In the Rips complex, all such simplices are edges; to increase or decrease a diagram coordinate value, one increases or decreases the length of the corresponding edge by pushing its two boundary points either further together or closer apart. To extend the example of Figure **??**, consider what would happen if the rightmost point were dragged further to the right: this would delay the creation of the larger cycle, and the corresponding point in the persistence diagram would go from $(2, \cdot)$ to $(2 + \delta, \cdot)$. The pairing of critical edges with coordinates in the diagram is well-defined locally, but non-infinitesimal perturbations of the metric space tend to shuffle around the critical pairings and change which edges are critical. There are also degenerate situations in which the pairing of edges with coordinate values in the diagram is not uniquely defined – these cases can be addressed theoretically, but do not concern us here as they are a measure-zero phenomenon. This differentiability of persistent homology allows us to use gradient descent methods for topological optimization [CCG$^+$21].

It is also important to note that persistent homology is not an injective invariant of metric spaces, meaning that different spaces can have the same persistence diagrams [Cur18, LT21, LHP21]. Thus, some information is lost in the process of converting a metric space to a persistence diagram.

Our focus in this paper is the task of computing functional sublevelset persistence. This is a very general invariant that can be computed for *constructible functions* on *definable sets*, which includes monotonic functions on simplicial complexes. However, we specialize here to the simple, ubiquitous setting where the domain is a cubical grid. This is not strictly necessary for the theory developed here, but significantly simplifies the analysis and computational pipeline, and encompasses many of the applications of interest.

To be precise, let $P \subset \mathbb{Z}^d$ be a rectangle inside of the integer lattice, and let $f : P \to \mathbb{R}$ be a function defined on P. P can be viewed as the vertex set of an m-dimensional cubical complex K_P^m in which higher-dimensional simplices are given by grid adjacencies, and f can be extended to this complex via the lower-* rule, in which $f(\sigma) = \max_{p \in \sigma} f(p)$, cf. [BGH$^+$21] section 4. Taking the *maximum* here guarantees that a simplex does not appear in a sublevelset filtration until all of its faces do, ensuring that the sublevelsets are valid subcomplexes.

There are many areas of application in which the topology of the pair (K_P^m, f) is useful. However, there are multiple challenges with computing and using these persistence diagrams:

- Persistent homology calculations scale poorly in the size of the input complexes, and hence cannot be effectively applied to many high-resolution images.
- Persistent homology calculations are sensitive to outliers, and many real-world images exhibit considerable noise.
- Even in the absence of noise, the topological structure in an image may not be evident without some image preprocessing.
- Different forms of preprocessing will produce different topological structures in the resultant image. How should we decide which is most relevant for the task at hand?
- Functional persistence is not injective, so different images can have the same persistence diagram. This implies a loss of information in any framework that makes decisions on the basis of these diagrams.

In this paper, we propose a framework for computing persistence of pairs (K_P^m, f) that addresses all of these concerns simultaneously. This framework is adapted from the concept of a *convolution layer* in modern deep learning. Let $B \subset \mathbb{Z}^d$ be another rectangle, smaller than P, and let $g : B \to \mathbb{R}$ be a function on this region; the pair (B, g) acts as a *convolutional filter*. Fix a vector $k = (k_1, \cdots, k_d)$ with $k_i \in \mathbb{N}_{>0}$, corresponding to the *stride* of the convolution. For $v \in \mathbb{Z}^d$, define:

$$(g * f)(v) = \sum_{p \in B} g(p) f(p + v \odot k),$$

where $\odot$ is the Hadamard product.Let $R \subset \mathbb{Z}^d$ be the collection of values r such that $B + (r \cdot k) \subseteq P$, which is necessarily also a rectangle. The pair $(R, g * f)$ is the output of our convolution. R may be much smaller than the support of f, since the various translates of B covering this support are not required to overlap much, if at all. In the degenerate setting where P and B are the same shape, R will consist of a single vertex. For the purposes of computing persistence, we will think of R, like P, as being the vertex set of a cubical complex K_R^m, and we extend functions on R to the entirety of the cubical complex using the lower-* rule.

By analogy with [SWB21b, SWB21a], we propose that the collection of persistence diagrams of the form $PH(K_R^m, g_i * f)$, for some set of filter functions $\mathcal{G} = \{g_i\}$, is of greater general utility than $PH(K_P^m, f)$. The computational advantages are immediate: when the stride k is large, so that R is much smaller than P, we have replaced a single, very expensive calculation with multiple, significantly faster calculations that can be performed in parallel. Moreover, we claim that this approach, which we deem *convolutional*

persistence, is more robust and flexible than ordinary persistence, and has superior inverse properties. It is also important to note that the *translation equivariance* of convolutions and the *transational invariance* of persistent homology work well together.

Stability and robustness are not hard to justify. If g is a filter function with $\|g\| \leq 1$, then $\|g*f_1 - g*f_2\| \leq \|f_1 - f_2\|$ by the Cauchy-Schwarz inequality, providing stability. A classical example is the *box-smoothing* filter $g(p) = \frac{1}{|B|}$. Such smoothing filters also do a good job of diminishing the impact of outliers. *Flexibility* is also immediate, since the collection of filters $\mathcal{G}$ can be curated for the task at hand, and can be learned using a training dataset.

Let us now demonstrate injectivity. For a fixed function $f : P \to \mathbb{R}$ and stride vector k, we can obtain a mapping ι_f of the rectangle R into $\mathbb{R}^{|B|}$ by sending every point $r \in R$ to the vector $\{f(b + k \odot r) \mid b \in B\}$. This is technically a set, rather than a vector, but it becomes a vector after fixing an order on the elements of B. Such a mapping can be extended linearly to the entire cubical complex K_R built on top of R. Let us suppose that ι_f is injective on the m-skeleton K_R^m, so that $\iota_f(K_R^m)$ has the structure of a simplicial complex isomorphic to K_R^m. Then for a function $g : B \to \mathbb{R}$, viewed also as a vector $\vec{g} \in \mathbb{R}^{|B|}$, and threshold value t, the sets $\{\sigma \in K_R^m \mid g(\sigma) \leq t\}$ and $\{\sigma \in \iota_f(K_R^m) \mid \sigma \cdot \vec{g} \leq t\}$ are homeomorphic. See Figure II.4 for a visual schematic. This means that the persistence diagram $PH(K_R^m, g * f)$ is identical to the persistence diagram $PH(\iota_f(K_R^m), \langle g, \cdot \rangle)$. If g is varied over the unit sphere in $\mathbb{R}^{|B|}$, the set of diagrams $PH(\iota_f(K_R^m), \langle g, \cdot \rangle)$ is known as the *Persistent Homology Transform* which is known to be injective, see [TMB14, GLM18, CMT18]. Thus, the collection of persistence diagrams of the form $\{PH(K_R^m, g * f) \mid g : B \to \mathbb{R}\}$ is injective. Note that the distinction between using all functions $g : B \to \mathbb{R}$, or only normalized functions lying on the sphere, is immaterial, as persistent homology transforms trivially under rescaling of the height function, so these two invariants determine one another.

The above proof of injectivity for *convolutional persistence* relies on $\iota_f : K_R^m \to \mathbb{R}^{|B|}$ being injective. We claim that this is generically the case whenever $\kappa = \Pi_i k_i > 2m$. Indeed, let $B^* \subseteq B$ consist of those elements in the top-left $k_1 \times k_2 \times \cdots k_d$ corner of B. The various translates $B^* + (k \odot r)$ are all disjoint subsets of P. We can thus view the vertices $\{f(b + k \odot r) \mid b \in B\}$ of $\iota_f(K_R^m)$ as having at least κ degrees of freedom, and this can be made more precise by projecting the embedding on to the coordinates corresponding to B^*. We now want to show that if we choose a collection S of points in $\mathbb{R}^M$ for $M > 2m$, and build the complete m-dimensional complex on top of these points, we will generically never have nontrivial intersections of disjoint simplices. To see why this is the case, let σ and τ be simplices corresponding to disjoint subsets S_σ and S_τ

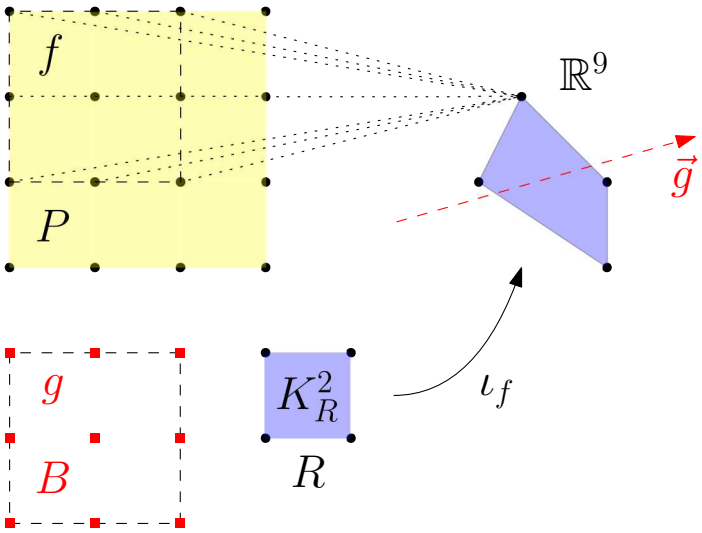

Fig. II.4: The function f is defined on the 4×4 grid on the top left. The box B is 3×3, and using a $(1, 1)$-stride there are four ways of laying B over the domain of f, so that R is a 2×2 grid. We can map the vertices of R into $\mathbb{R}^9$ by associating each vertex of R with its corresponding translate of B, and then taking as coordinates the values of f in that translate. This extends linearly to a map ι_f from the complex K_R^2 in $\mathbb{R}^9$, which here is shown to be an embedding.

of S. If σ and τ intersect, then their union is contained in an affine subset of $\mathbb{R}^M$ of dimension $(\dim \sigma + \dim \tau)$. Now, whenever $M > \dim \sigma + \dim \tau$, a collection of at least $(\dim \sigma + \dim \tau + 2)$ points generically does not lie on an affine subset of dimension $(\dim \sigma + \dim \tau)$. Since $M > 2m \geq \dim \sigma + \dim \tau$, and $|S_\sigma \cup S_\tau| = |S_\sigma| + |S_\tau| = (\dim \sigma + 1) + (\dim \tau + 1) = \dim \sigma + \dim \tau + 2$, we see that the intersection of σ and τ is not generic.

What we have just shown is that having a large stride vector, in addition to providing computational speedups by lowering the resolution of the resultant grid, also provides generic injectivity for persistence of higher-dimensional data complexes.

Now, what we have described above is persistence with a single convolutional layer. One can extend this construction by convolving the collection $\{(R, f * g) \mid g \in \mathcal{G}\}$ using a new rectangle B' and collection $\mathcal{H}$ of filter functions $h : B' \to \mathbb{R}$, and *then* compute persistence. This pipeline remains injective, because the collection $\{(R, g * f) \mid g : B \to \mathbb{R}\}$ determines the function f (this is trivial, consider characteristic functions of pixels in B), so the two-layer convolutional persistence is a composition of two injective operations, where we are now taking the sets $\mathcal{G}$ and $\mathcal{H}$ to consist of *all possible filters*. We might also consider introducing a non-linear activation between the convolutional layers, following more closely the structure of a CNN.

An alternative convolutional pipeline is to do a convolutional with a small stride, so that the resulting image has the same resolution as the original, and then apply a pooling later. This mirrors more closely the structure of a modern CNN. Note that this pipeline also enjoys an inverse result, provided that the corresponding embedding is injective on the input cubical complex.

In practice, of course, we are not interested in considering infinitely many filter functions. Indeed, learning tasks require invariants to *retain* features that are important to classification or regression, and *forget* features that are not. Thus, we will want to pick $\mathcal{G}$ appropriately. There are many ways for this choice to be made:

1) Take $\mathcal{G}$ to consist of a collection of popular filters in image processing, like blurring, sharpening, and boundary detection.
2) Take $\mathcal{G}$ to consist of Klein filters, as identified by Carlsson et al.
3) Take $\mathcal{G}$ to consist of eigenfilters identified via PCA on the set of patches of images in the training set.
4) Incorporate convolutional persistence in a deep learning pipeline and learn $\mathcal{G}$ to minimize a chosen loss function.

For the third point above, note that the inverse results for the PHT do not require all sphere directions when the shape in question sits inside a low-dimensional space of Euclidean space. Thus, it makes sense to ignore filters orthogonal to the space of image patches in question.

Prior Work: This is not the first work where convolutions and persistence have been investigated. They were considered in APE I. They are also mentioned in "PLLay: Efficient Topological Layer based on Persistence Landscapes". What makes this work unique is that it extends APE I by considering general convolutional layers, rather than noisy box smoothing, and it extends the PPLay paper by considering non-deep-learning approaches to convolutional persistence, focusing on the ability of convolutionals to speed up persistence calculations, rather than just extract more features for a CNN, and investigating more deeply into questions of injectivity and optimization.

Goal: The goal of this paper is to verify or rule out the following conjectures:

- Convolutional persistence can achieve reasonable accuracy on complex learning tasks for shapes defined by height functions over cubical grids.
- Convolutional persistence can be incorporated into other deep learning pipelines and provide some improvement in accuracy.

Experimental Pipeline:

1) In the first phase of the pipeline, we consider the digits dataset, which is small enough that ordinary persistence calculations are very fast. Thus, the role of this phase is to understand how convolutions improve the utility of persistence features. The first step of this phase is to see how standard ML models perform on digit classification based on cubical persistence features. We can then enrich this featurization using different choices of fixed filters $\mathcal{G}$, as outlined above, and see the impact this has on performance. The fundamental question here is whether or not reasonably good performance can be achieved through convolutional persistence.
2) The second phase of the pipeline is to try and learn the filters $\mathcal{G}$ using a deep learning architecture. This will require more work than choosing fixed filters as above.
3) The third phase is to extend the experiments above to larger data sets, like CIFAR, where the downsampling aspect of the convolutions will also be of importance.

REFERENCES

[BCD+14] Mickaël Buchet, Frédéric Chazal, Tamal K Dey, Fengtao Fan, Steve Y Oudot, and Yusu Wang, *Topological analysis of scalar fields with outliers*, arXiv preprint arXiv:1412.1680 (2014). ↑4

[BGH+21] Bea Bleile, Adélie Garin, Teresa Heiss, Kelly Maggs, and Vanessa Robins, *The persistent homology of dual digital image constructions*, arXiv preprint arXiv:2102.11397 (2021). ↑3, 4

[Car09] Gunnar Carlsson, *Topology and data*, Bulletin of the American Mathematical Society **46** (2009), no. 2, 255–308. ↑2

[CCG+21] Mathieu Carriere, Frédéric Chazal, Marc Glisse, Yuichi Ike, and Hariprasad Kannan, *Optimizing persistent homology based functions* (2021). ↑4

[CMT18] Justin Curry, Sayan Mukherjee, and Katharine Turner, *How many directions determine a shape and other sufficiency results for two topological transforms*, arXiv preprint arXiv:1805.09782 (2018). ↑5

[CSEH07] David Cohen-Steiner, Herbert Edelsbrunner, and John Harer, *Stability of persistence diagrams*, Discrete & computational geometry **37** (2007), no. 1, 103–120. ↑4

[Cur18] Justin Curry, *The fiber of the persistence map for functions on the interval*, Journal of Applied and Computational Topology **2** (2018), no. 3, 301–321. ↑4

[EH10] Herbert Edelsbrunner and John Harer, *Computational topology: an introduction* (2010). ↑2

[GHO16] Marcio Gameiro, Yasuaki Hiraoka, and Ippei Obayashi, *Continuation of point clouds via persistence diagrams*, Physica D: Nonlinear Phenomena **334** (2016), 118–132. ↑4

[Ghr08] Robert Ghrist, *Barcodes: the persistent topology of data*, Bulletin of the American Mathematical Society **45** (2008), no. 1, 61–75. ↑2

[GLM18] Robert Ghrist, Rachel Levanger, and Huy Mai, *Persistent homology and euler integral transforms*, Journal of Applied and Computational Topology **2** (2018), no. 1, 55–60. ↑5

[LCO18] Théo Lacombe, Marco Cuturi, and Steve Oudot, *Large scale computation of means and clusters for persistence diagrams using optimal transport*, Advances in Neural Information Processing Systems **31** (2018). ↑3

[LHP21] Jacob Leygonie and Gregory Henselman-Petrusek, *Algorithmic reconstruction of the fiber of persistent homology on cell complexes*, arXiv preprint arXiv:2110.14676 (2021). ↑4

[LT21] Jacob Leygonie and Ulrike Tillmann, *The fiber of persistent homology for simplicial complexes*, arXiv preprint arXiv:2104.01372 (2021). ↑4

[MMS11] Nikola Milosavljević, Dmitriy Morozov, and Primoz Skraba, *Zigzag persistent homology in matrix multiplication time*, Proceedings of the twenty-seventh annual symposium on computational geometry, 2011, pp. 216–225. ↑3

[OPT+17] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington, *A roadmap for the computation of persistent homology*, EPJ Data Science **6** (2017), 1–38. ↑3

[Oud15] Steve Y Oudot, *Persistence theory: from quiver representations to data analysis*, Vol. 209, American Mathematical Society Providence, 2015. ↑2

[PSO18] Adrien Poulenard, Primoz Skraba, and Maks Ovsjanikov, *Topological function optimization for continuous shape matching*, Computer graphics forum, 2018, pp. 13–25. ↑4

[SWB21a] Elchanan Solomon, Alexander Wagner, and Paul Bendich, *From geometry to topology: Inverse theorems for distributed persistence*, arXiv preprint arXiv:2101.12288 (2021). ↑4

[SWB21b] Yitzchak Solomon, Alexander Wagner, and Paul Bendich, *A fast and robust method for global topological functional optimization*, International conference on artificial intelligence and statistics, 2021, pp. 109–117. ↑4

[TMB14] Katharine Turner, Sayan Mukherjee, and Doug M Boyer, *Persistent homology transform for modeling shapes and surfaces*, Information and Inference: A Journal of the IMA **3** (2014), no. 4, 310–344. ↑5