CONVOLUTIONAL PERSISTENCE

EP

1. Overview

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: Give examples!

- 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?

Date: May 19, 2022.

- 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 1.1 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 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.

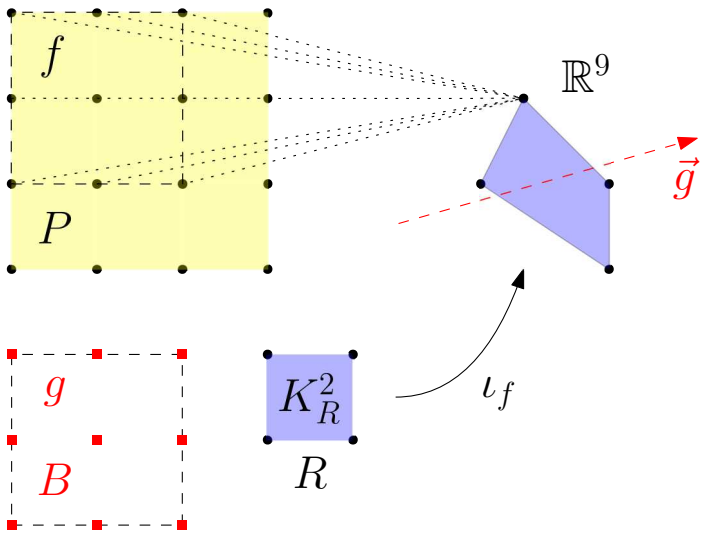

FIGURE 1.1. The function f is defined on the 4×4 grid on the top left. The box B is 3×3, and using a $(1,1)$-stride there are four ways of laying B over the domain of f, so that R is a 2×2 grid. We can map the vertices of R into $\mathbb{R}^9$ by associating each vertex of R with its corresponding translate 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.

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.

Make rigorous

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

[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). ↑1

[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). ↑1

[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. ↑1

[SWB21a] Elchanan Solomon, Alexander Wagner, and Paul Bendich, *From geometry to topology: Inverse theorems for distributed persistence*, arXiv preprint arXiv:2101.12288 (2021). ↑1

[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. ↑1

[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. ↑1