Persistent Homology through Image Segmentation (Student Abstract)

Joshua Slater[1], **Thomas Weighill**[2]

University of North Carolina at Greensboro
Department of Mathematics & Statistics, 116 Petty Building, PO Box 26170
Greensboro, NC 27402-6170
jwslater@uncg.edu[1], t_weighill@uncg.edu[2]

Abstract

The efficacy of topological data analysis (TDA) has been demonstrated in many different machine learning pipelines, particularly those in which structural characteristics of data are highly relevant. However, TDA's usability in large scale machine learning applications is hindered by the significant computational cost of generating persistence diagrams. In this work, a method that allows this computationally expensive process to be approximated by deep neural networks is proposed. Moreover, the method's practicality in estimating 0-dimensional persistence diagrams across a diverse range of images is shown.

Motivation

Recent work (de Surrel et al. 2022) has demonstrated the ability of deep neural networks to generate plausible vectorizations of persistence diagrams in both synthetic and practical settings. This work differs in three key respects, namely:

1. Computation is performed over grayscale image data, not 3D point clouds.
2. The model is tasked with identifying exact points in a persistence diagram, not generate plausible approximations of vectorized persistence diagrams.
3. Rather than require the use of a bespoke model architecture, by framing identification of persistence diagram points as an image segmentation problem, a variety of existing semantic segmentation architectures may be used.

Persistent Homology

Broadly, persistent homology identifies the scales at which topologically relevant features (connected components, holes, and higher-dimensional analogues) are created and destroyed. For a given grayscale image $\mathcal{I}$, these scales take the form of all pixel intensities $0 \leq t_1 < t_2 < \cdots < t_k \leq 1$ present within $\mathcal{I}$, giving rise to a sublevel set filtration (Figure 1)

$$K^{t_1} \subseteq K^{t_2} \subseteq \cdots \subseteq K^{t_k}$$

of binary images K^{t_a}, where

$$K^{t_a}_{i,j} = \begin{cases} 1 & \mathcal{I}_{i,j} \leq t_a \\ 0 & \text{else} \end{cases}$$

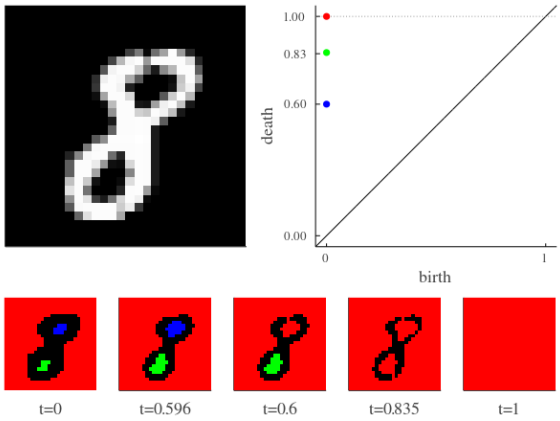

Figure 1: Example image (LeCun, Cortes, and Burges 2010) (top left), its corresponding 0-dimensional persistence diagram (top right) and sublevel set filtration (bottom).

Persistence diagrams (Figure 1) provide a succinct representation of structural changes that occur as the threshold increases. Each point (b, d) of a diagram indicates the thresholds at which a topological feature is born (b) and dies (d). In dimension zero, these features are connected components which begin with a single pixel (the *birth pixel*) and die when they are merged into an older component by the introduction of a new pixel (the *death pixel*). The lifetime of a topological feature, $d - b$, is a characterization of the feature's significance; features with long lifetimes are said to be more topologically significant than those with short lifetimes, which can often be attributed to noise.

Methods

Creation of Model Targets

Prior to training, the training dataset is thoroughly shuffled and partitioned into batches of size 16. The training process is then defined as follows (Figure 2):

1. A single random 32×32 patch is selected from each image in a batch. This further augments the training dataset, and ensures that the model is able to correctly identify persistence diagrams corresponding to a wide variety of images. These selections serve as the input to the model.

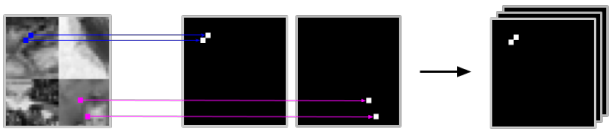

Figure 2: (Left to Right) A random 32×32 selection is made and true coordinates of topologically relevant pixels are noted, target arrays are created and then aggregated.

2. The true persistence diagrams of each 32×32 selection are computed using Ripserer (Čufar 2020).
3. For each point in the persistence diagram, we identify the birth pixel p_b and death pixel p_d for the corresponding connected component. We then encode these pixels in an array A defined by

$$A_{i,j} = \begin{cases} 1 & (i,j) \in \{p_b, p_d\} \\ 0 & \text{else} \end{cases}$$

These arrays, each of which represents a point of the true persistence diagram of the 32×32 pixel selection are treated as the model target. In practice, only the 10 points with the longest lifetime are processed in this manner.

Model Architecture and Loss

A variation of the Global Convolutional Network architecture (Peng et al. 2017) is chosen for its large receptive window, enabling distant birth and death pixels to be correctly identified and masked. To ensure the model is penalized both for falsely identifying pixels as topologically relevant and for failing to identify topologically relevant pixels, the dice loss,

$$L_{dice}(y, \hat{y}) = 1 - \frac{2\sum y\hat{y}}{\sum y^2 + \sum \hat{y}^2 + \epsilon}$$

is used in conjunction with the ADAM optimizer (Kingma and Ba 2015).

Results

To foster robust performance across a wide variety of grayscale images, training and testing datasets are comprised of a mixture of commonly used, publicly available image datesets: CIFAR10 (Krizhevsky and Hinton 2009) and UIUC Texture Dataset (Lazebnik, Schmid, and Ponce 2005). Images from the CIFAR10 dataset are converted to grayscale, then aggregated into larger images to account for discrepancies between the sizes of images across datasets.

The performance of the model on unseen data is evaluated by calculating the fraction of the total number of persistence points correctly identified. On average, the model correctly identifies $65.4 \pm 15.6\%$ of the top ten persistence points. Furthermore, $70.0 \pm 15.5\%$ of model predictions are present within the complete true persistence diagram of the model input, which includes points outside of those with the 10 longest lifetimes.

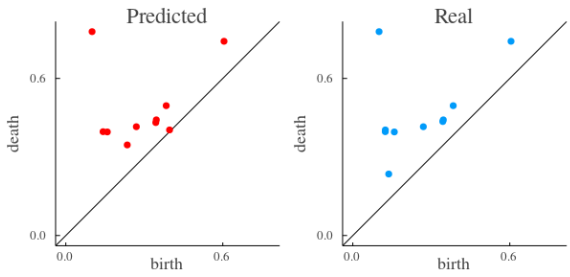

Figure 3: Example of a model identification with a bottleneck distance to the target diagram of 0.097.

We measure the discrepancy between model output and target persistence diagrams using the bottleneck distance, a standard metric in persistent homology. Overall, we find an average bottleneck distance of 0.099 ± 0.079 (Figure 3), suggesting that even erroneous model identifications provide reasonable approximations of the ground truth. Crucially, the computation time of the model greatly improves upon existing methods when processing large numbers of images (Table 1).

	Model	**Ripserer**
Computation Time: 128 Images	**3.5 ms**	63 ms

Table 1: Comparison of median computation time of the model vs. a state of the art persistent homology library.

References

de Surrel, T.; Hensel, F.; Carrière, M.; Lacombe, T.; Ike, Y.; Kurihara, H.; Glisse, M.; and Chazal, F. 2022. RipsNet: a general architecture for fast and robust estimation of the persistent homology of point clouds. *CoRR*, abs/2202.01725.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Lazebnik, S.; Schmid, C.; and Ponce, J. 2005. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8): 1265–1278.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Peng, C.; Zhang, X.; Yu, G.; Luo, G.; and Sun, J. 2017. Large Kernel Matters - Improve Semantic Segmentation by Global Convolutional Network. *CoRR*, abs/1703.02719.

Čufar, M. 2020. Ripserer.jl: flexible and efficient persistent homology computation in Julia. *Journal of Open Source Software*, 5(54): 2614.