Deep Learning with Topological Signatures

Romain Egele & Martin Cepeda

{romain.egele,martin.cepeda}@polytechnique.edu

November 2, 2020

Outline

Introduction

The basis of Deep Learning I

Deep learning uses several stacked layers of **neurons** (generally implementing linear operations) followed by a non-linear activation function:

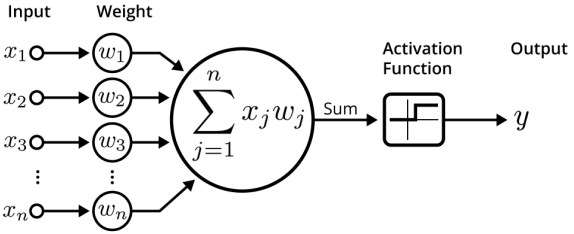


Figure: An artificial neuron (source)

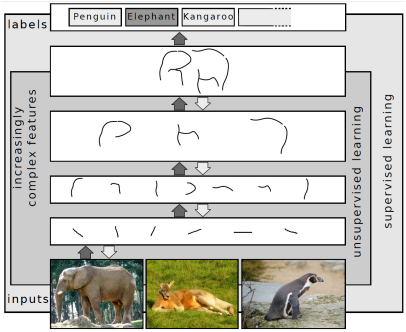


Figure: Feature learning process (source)

This stacking of non-linear operations is used for **feature learning** of the inputs the during a **training** stage to then perform predictions (classification/regression) during the **testing** stage.

The basis of Deep Learning II

In order to successfully train a neural network, several "ingredients" are necessary in terms of optimization:

- **Algorithm**: to minimize the loss of the model w.r.t. network parameters.
- (In most cases) **Layer backpropagation**: to perform gradient descent.
- **Regularization**: to control the complexity of a neural network model in order to avoid over-fitting.

Other aspects must also be taken into account (data harmonization, computing capabilities, initialization, callbacks during training, etc.) in order to train a network, which are out of the scope of this paper.

Persistence Diagrams I

Given a topological space X and a distance function f, a persistent diagram encodes the topological structure of the pair (X, f). More technically:

Simplex, complex, filtration (Boissonnat et al. [2020])

Topological spaces are represented by simplicial complexes. Let $V = \{1, \cdots, |V|\}$ be a set of vertices. A simplex σ is a subset of vertices $\sigma \subseteq V$. A **simplicial complex** K on V is a collection of **simplices** $\{\sigma\}, \sigma \subseteq V$, such that $\tau \subseteq \sigma \in K \Rightarrow \tau \in K$. The dimension $n = |\sigma| - 1$ of σ is its number of elements minus 1. A **filtration** of a simplicial complex is a function $F : K \to \mathbb{R}$ satisfying $F(\tau) \leq F(\sigma)$ whenever $\tau \subseteq \sigma$.

A filtration is done over sub-level sets of the distance function $f^{-1}((-\infty, t])$ for t ranging over $\mathbb{R}$. For a given dimension, the topological features span (birth and death) over the varying t mentioned before, which is encoded by a finite set of intervals: a **barcode**.

Persistence Diagrams II

Persistence diagram (Hofer et al. [2017] and Turner et al. [2014])

Let $\mathbb{R}^2_\star = \{(x_0, x_1) \in \mathbb{R}^2 : x_1 > x_0\}$. We define the k-th persistence diagram corresponding to the filtration F to be the multi-set of points in $\mathbb{R}^2_\star$ such that alongside countably infinite copies of the diagonal such that the number of points (counting multiplicity) in $[-\infty, a] \times [b, \infty]$ is equal to the number of topological features of dimension k that are born at or before a and die at or after b.

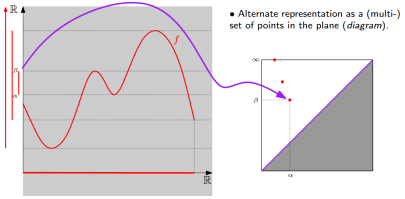


Figure: Encoding of a barcode into the plane (source). We call **persistence** of a feature its time span.

We note by $\mathbb{D}$ the set of all persistence diagrams with a finite quantity of non-essential features (those with finite persistence).

Persistence Diagrams for Deep Learning?

Neural networks take inputs from a vector space. In order to feed NNs with persistence diagrams, we need to have a mapping $T : \mathbb{D} \to \mathbb{R}^{n \times m}$. In order to find such a mapping:

- How to compute persistence diagrams from heterogeneous data types?
- How to define T?

This "topological step" allows to feed the neural-network model with topological, structural information of the input.

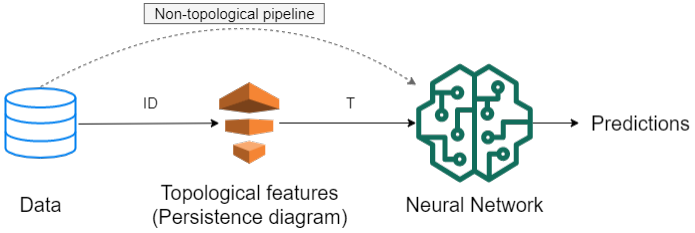


Figure: Deep Learning with Topological Signatures

Deep Learning with Topological Signatures

Contributions of the paper I

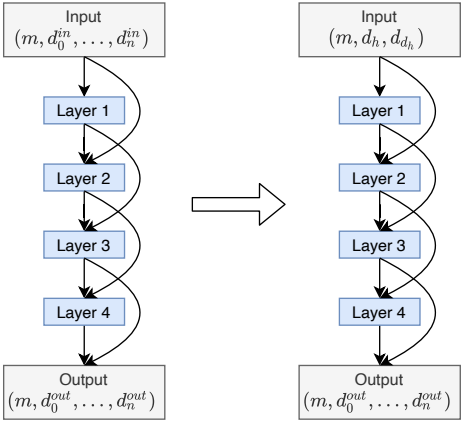


Figure: Vector inputs (left) and persistence diagram inputs (right) neural network architectures.

Contributions of the paper II

Hofer et al. [2017] introduce a novel neural network layer to **learn a mapping** $T : \mathbb{D} \to \mathbb{R}^N$. Given a persistence diagram $\mathcal{D} \in \mathbb{D}$ with x axis α the birth time and y axis β the death time:

1. Rotate the diagram by $-\frac{\pi}{4}$: x axis now represents $\alpha + \beta$ and y axis is $\beta - \alpha$ (persistence)
2. We note $\mathbb{R}^2_\Delta = \{(x_0, x_1) \in \mathbb{R}^2 : x_1 = x_0\}$ the diagonal in $\mathbb{R}^2$. Let $s : \mathbb{R}^2_\star \cup \mathbb{R}^2_\Delta \to \mathbb{R}^+_0$ Lipschitz continuous w.r.t. $\|\cdot\|_q$ and constant K_s and $s(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \mathbb{R}^2_\Delta$. In particular:

$$s(\boldsymbol{x}) = s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}(x_0, x_1) = \begin{cases} \exp\left[-\sigma_0^2(x_0 - \mu_0)^2 - \sigma_1^2(x_1 - \mu_1)^2\right] & x_1 \in [\nu, \infty) \\ \exp\left[-\sigma_0^2(x_0 - \mu_0)^2 - \sigma_1^2(\ln(\frac{x_1}{\nu})\nu + \nu - \mu_1)^2\right] & x_1 \in (0, \nu) \\ 0 & x_1 = 0 \end{cases}$$

   where ν is fixed and $\boldsymbol{\mu}, \boldsymbol{\sigma}$ are **trainable**. The mapping of $\mathcal{D}$ is the sum of $s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}(x_0, x_1)$ over all points in the rotated persistence diagram (visualization of s available at https://tda-layer-viz.herokuapp.com/)
3. Concatenate over a fixed number N of pairs $(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)_{i=0}^{N-1}$, pass to next layer

Contributions of the paper III

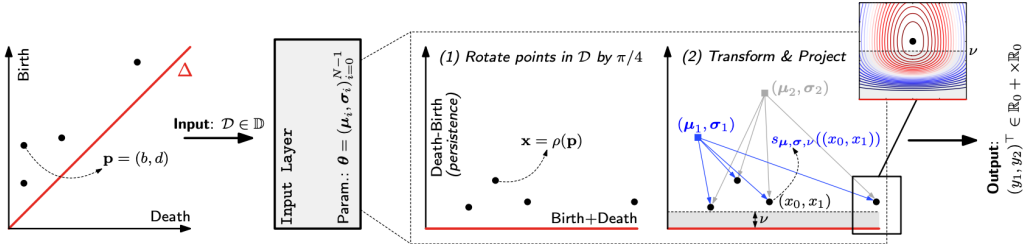


Figure: TDA input layer

Contributions of the paper IV

The previous pipeline makes sense because the proposed layer a) is stable w.r.t. the 1-Wasserstein distance w_1^q (robust to noise) b) has absolutely bounded first-order partial derivatives w.r.t. x_0 and x_1 on $\mathbb{R} \times \mathbb{R}^+$ and c) in particular, $s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}(\boldsymbol{x})$ is Lipschitz continuous with respect to[1] w_1^q on $\mathbb{D}$ (Lemmas 1, 2 and Theorem 1, respectively).

⚠ The particular definition of $s_{\boldsymbol{\mu},\boldsymbol{\sigma},\nu}(\boldsymbol{x})$ is trainable via **backpropagation** as it is differentiable w.r.t. $\boldsymbol{\mu}, \boldsymbol{\sigma}$ (sums and concatenations preserve differentiability).

⚠ The proposed layer (learned $\mathbb{D} \to \mathbb{R}^N$ mapping) achieves same kind of **stability** as fixed, learning task-dependent mappings.

[1] The q-Wasserstein distance is defined as the minimal value achieved by a perfect matching between the points of the two diagrams ($+$ all diagonal points), where the value of a matching is defined as the q-th root of the sum of all edge lengths to the power q. Edge lengths are measured in norm p, for $1 \leq p \leq \infty$ (Boissonnat et al. [2020])

List of experiments

Classification of 2D object shapes (2D arrays):

- *Animal*: 20 animal shapes classes, 100 samples each, 2,000 samples
- *MPEG-7*: 70 classes of object/animal shapes, 20 samples each, 1,400 samples

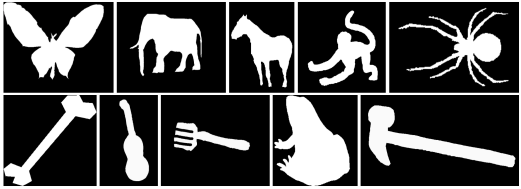


Figure: Animal (top) and MPEG-7 (bottom) data sets.

Classification of social networks graphs (unlabeled vertices, undirected edges):

- *reddit-5k*: 5 classes, 5,000 graphs.
- *reddit-12k*: 11 classes, about 12,000 graphs.

each sample represents a discussion graph and classes indicate subreddits (e.g., worldnews, video, etc.)

Classification of 2D object shapes

From a 2D array, we create a persistence diagram which we map to a vector space to feed it to a classifier.

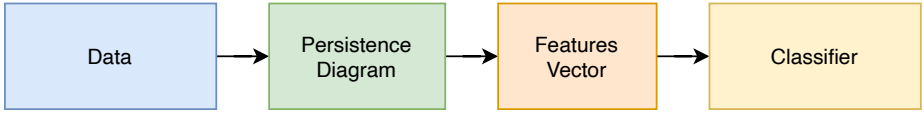


Figure: TDA + ML steps

Height function filtration (PHT) I

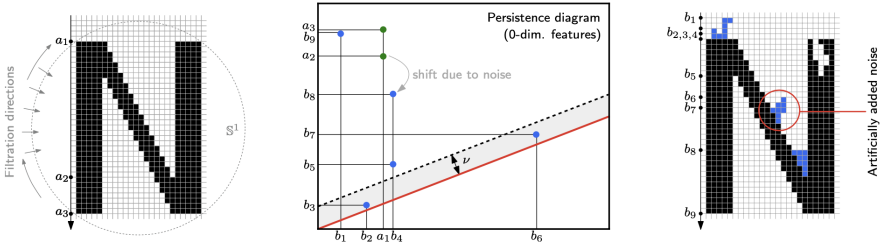
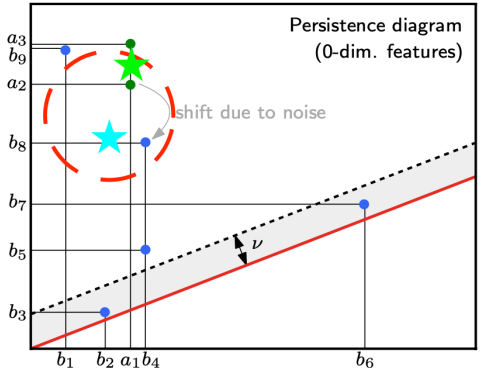


Figure: Height function filtration in the direction $d = (0, -1)$ of a clean (left) and noisy (right) shape.

Not invariant by rotation but invariant by translation!

Height function filtration (PHT) II



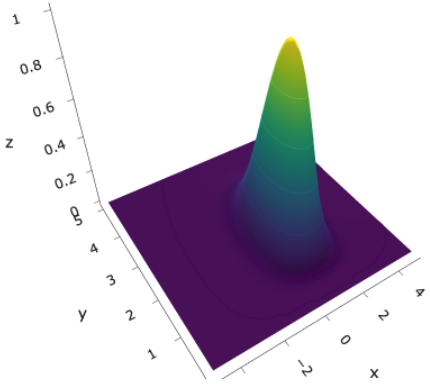


Figure: "Recognize an N" learned s

Examples of height function filtration I

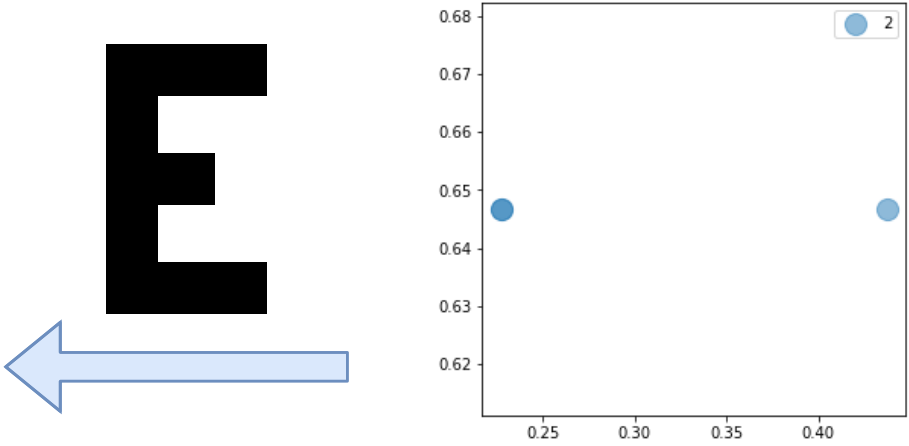


Figure: Height function filtration in the direction $d = (-1, 0)$ of a E shape.

Examples of height function filtration II

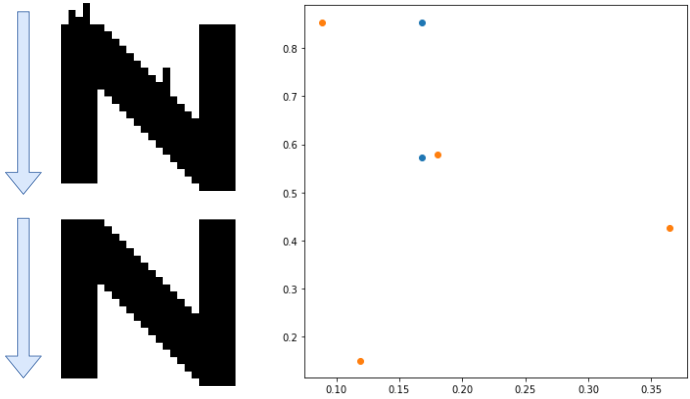


Figure: Reproduction of paper results with the N shape.

Temporal Performance of PHT

For 20 classes, 1 sample per class, 1 direction for PHT.

- 1 process: 46 sec.
- 8 processes: 10 sec.

Good and easy parallelism! Thank to parallelization we managed to generate the PHT with 1, 2 and 16 directions for the *Animal* data set. It took about 5 hours for the computation of 16 directions using 10 parallel processes.

Vectorization of Persistence Diagram

•Idea: directly use the persistence diagrams as features via vectorization

Algorithm 1: Vectorization of Persistence Diagram

Input: $\mathcal{D}$ persistence diagram, N output vector size
Result: vector V of size N
tmp = list()
for (b, d) *in* $\mathcal{D}$ **do**
    p = d - b /* persistence */
    tmp.append(p)
end
tmp.sort(reverse=True) /* decreasing order */
if *leng(tmp)* $> N$ **then**
    V = tmp[:N]
else
    V = tmp + [0]*(N-len(tmp))
end

Baseline performance I

•Idea: Use the vetorization of persistence diagrams to directly fit a classifier.

	N						
	5	10	20	40	80	160	**Ours**
MPEG-7	81.8	82.3	79.7	74.5	68.2	64.4	**91.8**
Animal	48.8	50.0	46.2	42.4	39.3	36.0	**69.5**
reddit-5k	37.1	38.2	39.7	42.1	43.8	45.2	**54.5**
reddit-12k	24.2	24.6	27.9	29.8	31.5	31.6	**44.5**

Figure: Results against Baseline

N is the size of the vectorized persistence diagram (i.e., 10 directions with $N = 5$ gives an input vector of size 50). The baseline classifier used is a linear support vector machine (SVM). All the even directions of 32-PHT were used (i.e., 16 directions).

Baseline performance II

Classifiers	SVM	RF (Ours)	NN-1 (Ours)
Animal	50.0	56.0 ± 2.7	53.0

Table: Classification accuracies for different classifier. A linear support vector machine (SVM), random forest (RF), neural network (NN). Where the vectorized persistence diagram is of size N=10.

Neural Network Architecture I

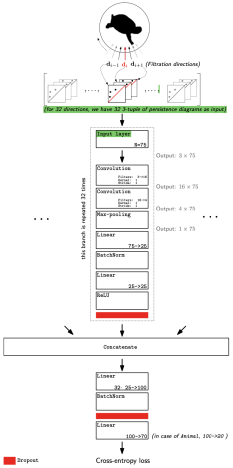


Figure: Neural architecture with TDA-layer

Neural Network Architecture II

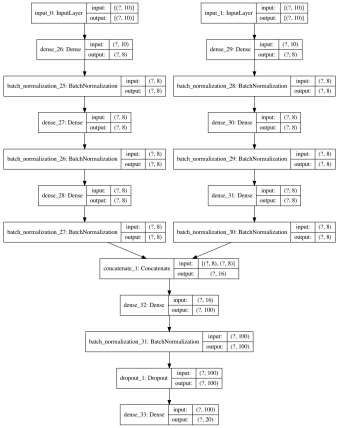


Figure: Fully connected neural architecture with multiple inputs.

Neural Network Architecture III

Classifiers	NN TDA	NN 1 input no TDA	NN N inputs no TDA
Test Accuracy	69.5	53.0	71.0

Table: Performance of classic fully connected neural networks and TDA-layer neural network.

Temporal performance:

- NN TDA: 8.6 sec per epoch for 500 epochs.
- N inputs no TDA: 0.37 sec per epoch for 100 epochs.

Conclusion

Limitations

- Prohibitive computing time for higher order features ($q \geq 2$, voids and higher order homology groups)
- **the claim "no specific data preprocessing" for `MPEG-7` and `Animal` datasets is, at least, non accurate**: Turner et al. [2014] mention that PHT (image $\rightarrow$ persistence diagram) centers, scales and rotates images (SOTA[2] in $\mathcal{O}(n^2)$) prior to computing projections
- It needs a high number of epochs (e.g., 500 epochs in the experiments)

[2] as of 2019: Rangan et al. [2019]

Possible extensions

- Adding image "channels" by pre-segmentating input image
- Application to time-series data (even sound?)
- Transfer learning (e.g. train on a "basic shapes" dataset and use learned embedding with a more high-level problem such as image classification)

References

References I

Boissonnat, J.-D., Glisse, M., Maria, C., Michel, B. and Rouvreau, V. [2020]. GUDHI library documentation, https://gudhi.inria.fr/python/latest/. Accessed: 2020-27-10.

Hofer, C. D. [2020]. torchph, https://github.com/c-hofer/torchph.

Hofer, C. D., Kwitt, R., Niethammer, M. and Uhl, A. [2017]. Deep learning with topological signatures, https://arxiv.org/abs/1707.04041.

Rangan, A., Spivak, M., Andén, J. and Barnett, A. [2019]. Factorization of the translation kernel for fast rigid image alignment, https://arxiv.org/abs/1905.12317.

Turner, K., Mukherjee, S. and Boyer, D. M. [2014]. Persistent homology transform for modeling shapes and surfaces, https://arxiv.org/abs/1310.1030.