Supplementary Material

Mathieu Carrière [1] **Frédéric Chazal** [2] **Marc Glisse** [2] **Yuichi Ike** [3] **Hariprasad Kannan** [2] **Yuhei Umeda** [3]

Proof of Proposition 3.2

Proposition. *Given a simplicial complex K, the map* $\mathrm{Pers}\colon \mathrm{Filt}_K \subseteq \mathbb{R}^{|K|} \to \mathbb{R}^{|K|}$ *is semi-algebraic, and thus definable in any o-minimal structure. Moreover, there exists a semi-algebraic partition of* Filt_K *such that the restriction of* Pers *to each element of this partition is a Lipschitz map.*

Proof. As K is finite, the number of preorders on the simplices of K is finite. Let $\preceq$ be a preorder on simplices of K induced by some equalities and inequalities between the coordinates of $\mathbb{R}^{|K|}$. Then, the set of filtrations $F \in \mathrm{Filt}_K$ such that F gives rise to a preorder equal to $\preceq$ is a semi-algebraic set. Thus, Filt_K is a semi-algebraic set that admits a semi-algebraic partition such that the restriction of the persistence map Pers to each component is a constant permutation. As a consequence, on each open element of this partition, the partial derivatives of Pers are equal to constant 0 or 1.

Now, from the stability theorem for persistence (Chazal et al., 2016), the persistence modules induced by two filtrations $F_1, F_2 \in \mathrm{Filt}_K$ are ϵ-interleaved where ϵ is the sup norm of the vector $F_2 - F_1$. As a consequence, the restriction of Pers to each component of the above semi-algebraic partition of Filt_K is 1-Lipschitz with respect to the sup norm in $\mathbb{R}^{|K|}$. □

Other examples of definable families of filtrations

The Čech and alpha-complex filtrations The Čech complex filtration built on top of sets of n points $x_1, \ldots, x_n \in \mathbb{R}^d$ is the semi-algebraic parametrized family of filtrations

$$\Phi\colon A = (\mathbb{R}^d)^n \to \mathbb{R}^{|\Delta_n|} = \mathbb{R}^{2^n-1},$$

where Δ_n is the simplicial complex made of all the faces of the $(n-1)$-dimensional simplex, defined, for any $(x_1, \ldots, x_n) \in A$ and any simplex $\sigma \subseteq \{1, \ldots, n\}$, by

$$\Phi_\sigma(x) = \min \left\{ r \geq 0 \ : \ \bigcap_{j \in \sigma} B(x_j, r) \neq \emptyset \right\}$$

where $B(x_j, r) = \{x \in \mathbb{R}^d : \|x - x_j\| \leq r\}$ is the ball with center x_j and radius r. The filtrations naturally satisfy the Lipschitz property.

Note that the Čech complex filtration is closely related to the so-called alpha-complex filtration which is a filtration of the Delaunay triangulation of the set of points $x_1, \ldots, x_n$—see, for example, Chapter 6 in (Boissonnat et al., 2018) for a definition. The simplicial complex on which the alpha-complex filtration is built depends on the points $x_1, \ldots, x_n$. However, if A is a connected component of the open semi-algebraic subset of $(\mathbb{R}^d)^n$ of the points $x_1, \ldots, x_n$ that are in general position[1], then all the points in A have the same Delaunay triangulation. In that case the alpha-complex defines a semi-algebraic parametrized family of filtrations.

Moreover, the persistence diagram of the alpha-complex filtration built on top of $x_1, \ldots, x_n \in \mathbb{R}^d$ is the same as the persistence diagram of the Čech complex filtration built on the same set of points (Edelsbrunner, 1993; Bauer & Edelsbrunner, 2017) if we ignore points on the diagonal.

Cubical complexes While this presentation was restricted to simplicial complexes for simplicity, the same properties apply for more general complexes. Indeed, as long as the boundary maps are well-defined and associate to each cell of dimension d a chain of dimension $d-1$, the same persistence algorithm can be run, and its output is still a permutation of its input. The most common non-simplicial complexes are the so-called cubical complexes, where cells are cubes. They are particularly well suited to represent images, or (discretized) functions on $\mathbb{R}^d$.

*Equal contribution [1]Université Côte d'Azur, Inria, France [2]Université Paris-Saclay, CNRS, Inria, Laboratoire de Mathématiques d'Orsay, France [3]Fujitsu Ltd., Kanagawa, Japan. Correspondence to: Mathieu Carrière <mathieu.carriere@inria.fr>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

[1]The points $x_1, \ldots, x_n \in \mathbb{R}^d$ are in general position if any subset of size at most $d+1$ is a set of affinely independent points, and if no subset of $d+2$ points lies on the same $(d-1)$-dimensional sphere

Other examples of locally Lipschitz functions of persistence

Persistence images (Adams et al., 2017) Given a weight function $w\colon \mathbb{R}^2 \to \mathbb{R}^+$ and a real number $\sigma > 0$, the persistence image (also called the persistence surface) associated with a persistence diagram D is the function $\rho_D\colon \mathbb{R}^2 \to \mathbb{R}$ defined by

$$\rho_D(q) = \sum_{p \in D} w(p) \exp\left(-\frac{\|q-p\|^2}{2\sigma^2}\right).$$

The definition of ρ_D only involves algebraic operations, the weight function w, and exponential functions. Thus, it follows from (Wilkie, 1996) that if w is a semi-algebraic function, K is a simplicial complex, and $\{q_1, \ldots, q_n\}$ is a finite set of points in $\mathbb{R}^2$, then the map that associates the vector $(\rho_D(q_1), \ldots, \rho_D(q_n))$ to each persistence diagram D of a filtration of K is definable in some o-minimal structure. In (Adams et al., 2017), the authors provide explicit conditions under which the map from persistence diagrams to persistence surfaces is Lipschitz.

A remark about the locally Lipschitz condition in Theorem 4.2

The convergence Theorem 4.2 provides explicit conditions ensuring the convergence of stochastic gradient descent for functions of persistence. The main criterion to be checked is the locally Lipschitz condition for $\mathcal{L}$. From the remark following Definition 4.1, it is sufficient to check that Φ and E are Lipschitz. Regarding Φ, it is obvious for Vietoris-Rips, Čech, DTM, etc. filtrations on finite point clouds, but wrong for the alpha-complex filtration, which is not a locally Lipschitz function of the coordinates of the points. Indeed, one can take 3 points within a bounded region of the plane that are almost aligned and whose circumradius is arbitrarily large, and this circumradius is the filtration value of the triangle. However, by comparing with the Čech complex, we know that such large values always correspond to diagonal points of the persistence diagram. In our example of 3 points, the longest edge has the same filtration as the triangle and they are paired by the persistence algorithm. To handle alpha-complex filtrations, we need to restrict to functions of persistence that are defined for various numbers of points, ignore points on the diagonal (the image of a diagram is the same if we add or remove points on the diagonal), and are still locally Lipschitz. This is the case for all the functions of persistence presented in this paper. Composed with such a function, the difference between Čech and alpha-complex filtrations disappears, it becomes an implementation detail and all the differentiation and optimization properties proved for the first apply to the second.

More numerical experiments

3D shape processing. In (Poulenard et al., 2018), persistence optimization is used for modifying functions defined on 3D shapes. More specifically, given a 3D shape S, one is interested in optimizing a function $F\colon V(S) \to \mathbb{R}$ defined on the vertices $V(S)$ of S, so that the Wasserstein distance between the persistence diagram associated with F and D^* is minimized, where D^* is a target persistence diagram (which either comes from another function $G\colon S \to \mathbb{R}$, or is defined by the user). In this experiment, we minimize the loss $\mathcal{L}(F) = T(F)$, where $T(F) := W_2(D, D^*)^2$, that is, the Wasserstein distance between the 0-dimensional persistence diagram D associated with the sublevel set of a function F—initialized with the height function, see Figure 5 (1st row)—of a human 3D shape, and a target persistence diagram D^* which only contains a single point, with the same coordinates than the point (in the persistence diagram of the height function of the shape S) corresponding to the right leg. This makes the function values to force the two points in D corresponding to the hands of S to go to the diagonal, by creating paths between the hands and the hips (2nd row). It is worth noting that these path creations come from the fact that we only used a topological penalty in the loss: in (Poulenard et al., 2018), the authors ensure smoothness of the resulting function by forcing it to be a linear combination of a first few eigenfunctions of the Laplace-Beltrami operator on the 3D shape. We also display the sequence of optimized persistence diagrams in Figure 5 (4th row), from which it can be observed that the optimization is piecewise-linear, which reflects the fact that the persistence map has an associated semi-algebraic partition, as per Proposition 3.2.

Image processing. Another task presented in (Brüel-Gabrielsson et al., 2020) is related to image processing. In this experiment, we optimize the 0-dimensional homology associated with the pixel values of a binary image I of a digit with noise (see Figure 6, upper left). Since noise can be detected as unwanted small connected components, we use the loss $\mathcal{L}(I) = P(I) + T(I)$, where $T(I) := \sum_{i=1}^{p} |d_i - b_i|$ is the total persistence penalty, $D_{\text{reg}} = \{(b_1, d_1), \ldots, (b_p, d_p)\}$ is the finite 0-dimensional persistence diagram of the cubical complex associated with I, and $P(I) := \sum_{p \in I} \min\{|p|, |1-p|\}$ is a penalty forcing the pixel values to be either 0 or 1. As can be seen from Figure 6, using both penalties is essential: if only $P(I)$ is used, the noise is amplified (upper right), and if only $T(I)$ is used, the noise does not quite disappear, and paths are created between the noise and the central digit to ensure the corresponding connected components are merged right after they appear (middle left). Note that this funny behavior that appears when penalizing topology alone is similar to what was observed in experiments where persistence was used to

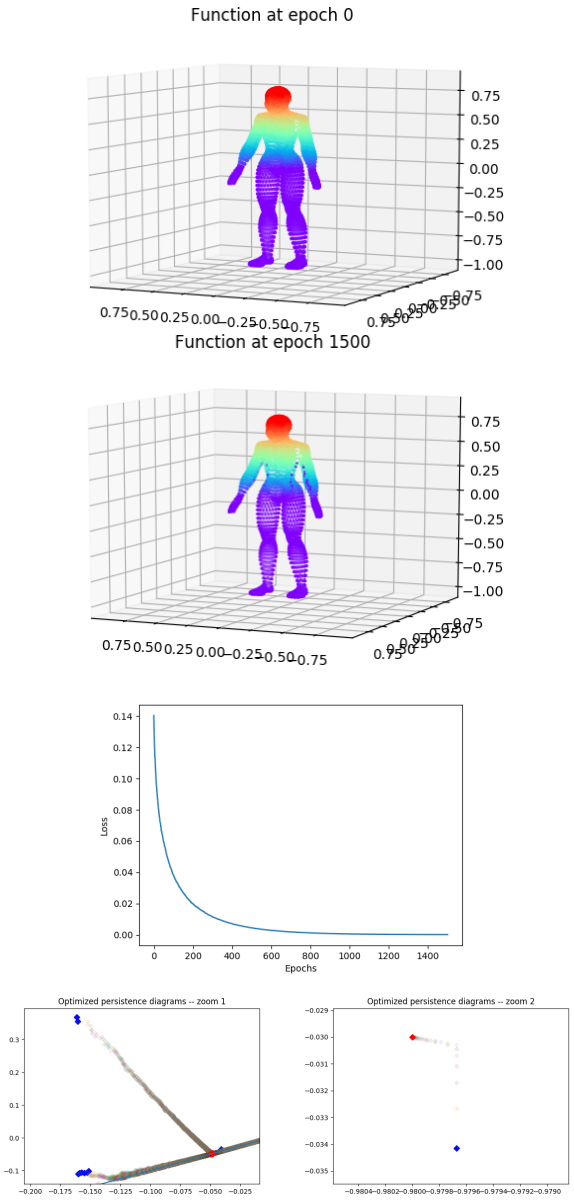

Figure 5. 3D shape before (1st row) and after (2nd row) optimization, and corresponding loss function (3rd row). Note how paths of low function values were created between the hips and the hands. We also show the sequence of persistence diagrams (4th row) with blue points being the initial persistence diagram, and red ones being the fully optimized persistence diagram. Note how optimization trajectories look piecewise-linear.

simplify functions (Attali et al., 2009). Using both penalties completely removes the noise (middle right) in two steps: firstly topology is optimized, and then the paths created by optimizing $T(I)$ are removed by optimizing $P(I)$. This two step process can also be observed on the loss function (lower left) and the sequence of optimized persistence diagrams of the image (lower right), where a bifurcation point can be observed in the optimization process.

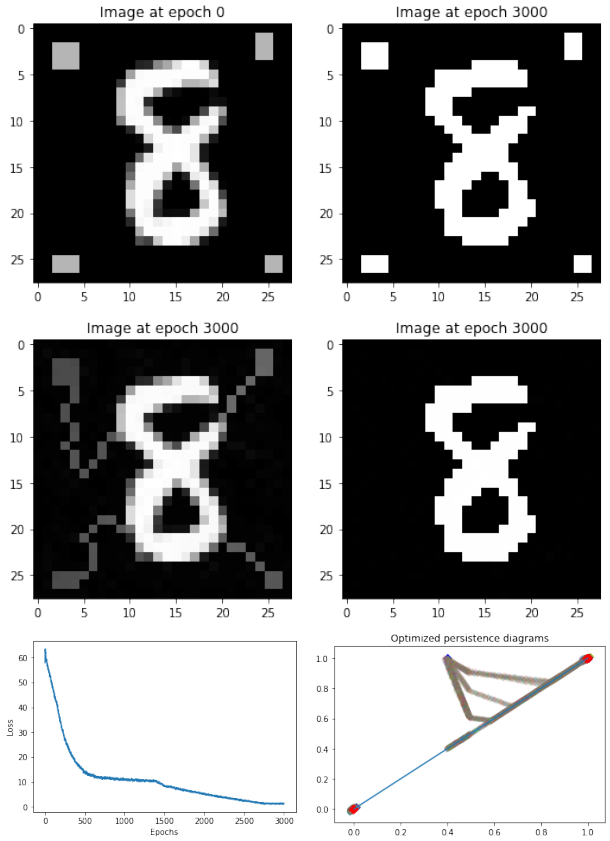

Figure 6. Image before (upper left) and after optimization for various penalties. For the case when both $T(I)$ and $P(I)$ are used (middle right), we also show the corresponding loss function (lower left) and the sequence of persistence diagrams (lower right). The blue points are the initial persistence diagram, and the red ones are the fully optimized persistence diagram.

Linear regression. This experiment comes from (Brüel-Gabrielsson et al., 2020), in which the authors use persistence optimization in a linear regression setting. Given some dataset $X \in \mathbb{R}^{n \times p}$ and ground-truth associated values $Y \in \mathbb{R}^n$ computed as $Y = X \cdot \beta^* + \epsilon$, where $\beta^* \in \mathbb{R}^p$ is the vector of ground-truth coefficients and ϵ is some high-dimensional Gaussian noise, one can leverage some prior knowledge on the shape of β to penalize the coefficients with bad topology. In particular, when using β^* with three peaks, as in Figure 7 (left), we use the loss $\mathcal{L}(\beta) = P(\beta) + TV(\beta) + T(\beta)$, where $T(\beta) := \sum_{i=1}^{p} |d_i - b_i|$, $\tilde{D} = \{(b_1, d_1), \ldots, (b_p, d_p)\}$ is the 0-dimensional persistence diagram of the sublevel sets of β, minus the three most prominent points, $TV(\beta) = \sum_i |\beta_{i+1} - \beta_i|$ is the usual total variation penalty (which can also be interpreted as a topological penalty as it corresponds to the total persistence of the so-called *extended persistence* (Cohen-Steiner et al., 2009) of the signal), and $P(\beta) := \sum_i |x_i \cdot \beta - y_i|^2$

is the usual mean-square error (MSE). We optimized β with the MSE alone, then MSE plus total variation, then MSE plus total variation plus topological penalty, and we generated new MSE values from new randomly generated test data, see Figure 7 (top). It is interesting to note that using all penalties lead to the best result: using MSE alone leads to overfitting, and adding total variation smooths the coefficients a lot since β is initialized with random values. Using all penalties ends up being a right combination of minimizing the error, smoothing the signal, and getting to the right shape of β.

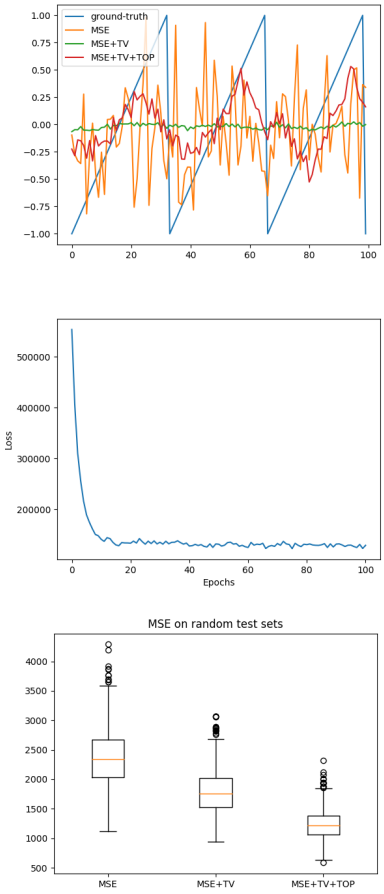

Figure 7. Regression coefficients after optimization for various penalties (1st row), and corresponding loss function when all penalties are used (2nd row). Generalization performance is increased by using all penalties, since the MSE on various randomly generated test sets is largely decreased (3rd row).

Noisy point cloud. We perform another point cloud optimization (as in the article), but now we start with a noisy sample X of the circle with three outliers and use the loss $\mathcal{L}(X) = W_2(D, D^*)^2$, where W_2 stands for the 2-Wasserstein distance, D is the 0-dimensional persistence diagram associated with the Vietoris-Rips filtration of X, and D^* is the 0-dimensional persistence diagram associated with the Vietoris-Rips filtration of a clean (i.e., with neither noise nor outliers) sample of the circle. See Figure 8. Note that when one does not use extra penalties, optimizing only topological penalties can lead to funny effects: as one can see on the middle of Figure 8, the circle got disconnected, and one of the outliers created a small path out of the circle during optimization.

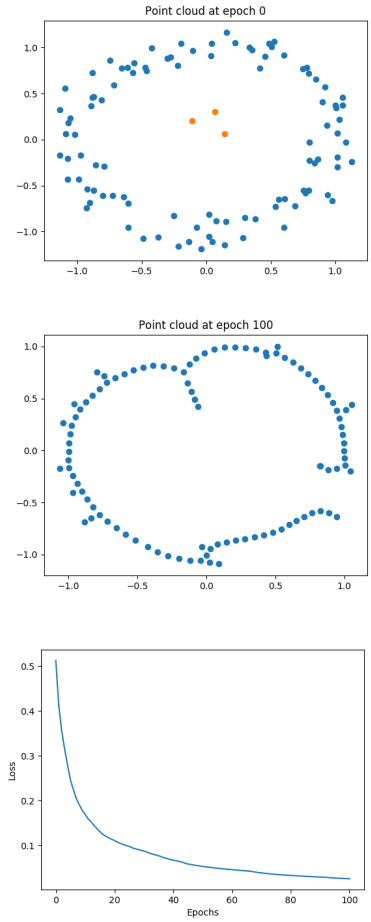

Figure 8. Noisy circle initialization with outliers, before (top) and after (middle) optimization, and corresponding loss function (bottom).

Filter optimization. In addition to classifying digits (as presented in the article), we also classify various graph data sets from TU Dortmund (Morris et al., 2020). More specifically, following (Hofer et al., 2020), we compute, for each graph, two sublevel set filtrations in homological dimension 0, using two filter functions defined on the graph vertices, and parametrized by a Graph Isomorphism Network (GIN) (Xu et al., 2019) with three MLP layers of sizes 32, 32 and 2 (initialized randomly) and ϵ parameter set to 0. We then optimize the parameters of these GINs (there is

one for each graph data set) using loss (3) in the article, and use the optimized parameters to produce two corresponding 0-dimensional persistence diagrams for each graph.

As with MNIST images, we then train random forests on these data sets of (pair of) persistence diagrams, by computing and concatenating their five first persistence landscapes with resolution 100, before and after optimization. A baseline is added by directly classifying the graphs with the eigenvalues of their graph Laplacians with a random forest classifier. The scores obtained for 10-fold cross-validation are displayed in Table 2. Note that, contrary to MNIST images, classifying graphs is not necessarily a binary classification task as some graph data sets have more than two associated classes. Similarly to MNIST images, optimizing the graph filters using loss (3) can drastically improve classification accuracy afterwards, and can be run without making any assumption on the structure of the graphs themselves, such as their number of nodes, edges, cycles, etc. On the other hand, any decrease in performance is always mild and reasonable.

The accuracy scores for all binary and multi-class tasks for MNIST images and graphs are provided in Table 2, and all pairwise scores for MNIST images are also shown in Figure 9.

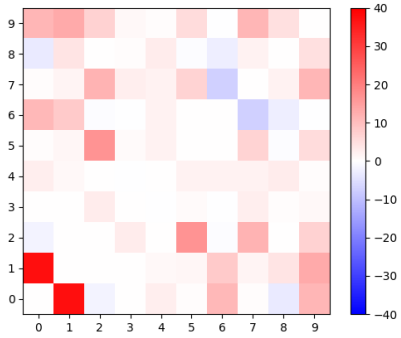

Figure 9. Pairwise differences between the accuracy scores (%) of optimized and non optimized filtrations for classifying digit x vs. digit y. One can see that the difference is almost always positive.

Details on experiments

All the experiments were run on a computer that has Intel dual-Xeon SP with 10 cores and 9.6GB/core (RAM). The classification results for the experiments on graphs were generated with 10-fold cross-validation: each data set was divided into ten 90-10 train-test splits, and results were averaged over these splits. Random forest parameters are the default ones of `Scikit-Learn`. The classification results for the experiments on images were generated using train-test splits provided in `TensorFlow`. We also added a multi-class classification task, called `all`, which consists of jointly classifying all images. All the optimization processes (including those presented in the article) were done with stochastic gradient descent as implemented in `TensorFlow` 2.4.1. More specifically, we used the `SGD` optimizer class with `InverseTimeDecay` learning rate (in order to satisfy Assumptions 1, 2 and 3 of Section 4.1 in the article). We also used `Adam` optimizer with `ExponentialDecay` learning rate for some experiments since we noticed empirically that, even though Assumption 1 was not satisfied, the results were not very different, and convergence was slightly smoother. Parameter initialization was done randomly, and the batch sizes and the numbers of epochs were taken sufficiently large so that convergence was reached in each illustration—see the code (`illustrations.ipynb` and `optim_filters.py`) for exact values. For instance, filter selection was done with initial learning rate equal to 0.001 and batch size equal to 150.

Dataset	Baseline	Before	After	Difference
all	96.9	65.1	80.6	**+15.5**
vs01	100.0	61.3	99.0	**+37.6**
vs02	99.4	98.8	97.2	-1.6
vs03	99.8	99.1	99.2	**+0.1**
vs04	99.9	96.0	98.8	**+2.8**
vs05	99.6	95.7	96.3	**+0.6**
vs06	99.4	87.3	98.2	**+10.9**
vs07	99.8	97.4	98.0	**+0.6**
vs08	99.4	90.4	87.0	-3.4
vs09	99.4	86.8	98.3	**+11.5**
vs12	99.6	98.3	98.5	**+0.2**
vs13	99.7	98.9	99.1	**+0.2**
vs14	100.0	97.1	98.3	**+1.2**
vs15	99.8	96.7	98.0	**+1.3**
vs16	99.7	89.0	97.3	**+8.3**
vs17	99.7	96.8	98.6	**+1.8**
vs18	99.8	91.7	96.0	**+4.3**
vs19	99.6	84.8	98.0	**+13.2**
vs23	99.1	95.2	98.0	**+2.9**
vs24	99.4	98.7	98.7	0.0
vs25	99.4	80.6	97.2	**+16.6**
vs26	99.7	98.8	98.2	-0.6
vs27	98.6	80.1	91.9	**+11.8**
vs28	99.1	96.8	96.8	0.0
vs29	99.1	91.6	98.6	**+7.0**
vs34	99.8	99.4	99.1	-0.3
vs35	99.2	93.5	94.3	**+0.8**
vs36	99.7	99.3	99.3	-0.1
vs37	98.9	94.9	97.5	**+2.6**
vs38	99.0	98.3	98.8	**+0.6**
vs39	98.8	96.8	97.8	**+1.0**
vs45	99.9	96.5	98.4	**+1.9**
vs46	99.6	94.1	96.0	**+1.9**
vs47	99.7	97.2	99.3	**+2.1**
vs48	99.2	90.4	93.4	**+3.0**
vs49	98.4	93.7	94.2	**+0.5**
vs56	99.0	96.9	97.1	**+0.2**
vs57	99.7	90.5	97.2	**+6.7**
vs58	98.9	92.7	92.3	-0.4
vs59	99.4	90.0	95.4	**+5.5**
vs67	99.7	98.4	91.0	-7.4
vs68	98.7	92.2	89.5	-2.7
vs69	99.7	87.0	86.7	-0.3
vs78	98.9	95.7	97.6	**+1.9**
vs79	99.1	85.3	96.9	**+11.5**
vs89	98.7	84.2	89.1	**+4.9**
PROTEINS	73.6 ± 3.2	67.6 ± 4.13	69.7 ± 4.63	**+2.1**
MUTAG	85.1 ± 7.1	82.9 ± 9.34	87.3 ± 8.23	**+4.4**
COX2	78.6 ± 1.7	74.1 ± 4.39	74.3 ± 3.60	**+0.2**
DHFR	78.8 ± 4.1	60.2 ± 12.33	62.5 ± 10.02	**+2.3**
BZR	84.9 ± 2.1	78.2 ± 6.83	76.0 ± 9.33	-2.3
FRANKENSTEIN	69.7 ± 1.4	59.7 ± 2.49	60.4 ± 1.14	**+0.7**
IMDB-MULTI	49.3 ± 3.3	36.9 ± 1.74	36.7 ± 1.87	-0.2
IMDB-BINARY	72.8 ± 4.5	61.7 ± 3.52	62.7 ± 4.75	**+1.0**
NCI1	74.3 ± 1.8	74.0 ± 3.30	72.3 ± 3.21	-1.8
NCI109	72.5 ± 1.7	73.6 ± 2.42	72.8 ± 1.62	-0.8

Table 2. All accuracy scores for graphs and MNIST data sets.

References

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., and Ziegelmeier, L. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(1):218–252, 2017.

Attali, D., Glisse, M., Hornus, S., Lazarus, F., and Morozov, D. Persistence-sensitive simplification of functions on surfaces in linear time. *TOPOINVIS*, 9:23–24, 2009.

Bauer, U. and Edelsbrunner, H. The morse theory of čech and delaunay complexes. *Transactions of the American Mathematical Society*, 369(5):3741–3762, 2017.

Boissonnat, J.-D., Chazal, F., and Yvinec, M. *Geometric and topological inference*, volume 57. Cambridge University Press, 2018.

Brüel-Gabrielsson, R., Nelson, B., Dwaraknath, A., Skraba, P., Guibas, L., and Carlsson, G. A topology layer for machine learning. In *23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, pp. 1553–1563. PMLR, 2020.

Chazal, F., de Silva, V., Glisse, M., and Oudot, S. *The structure and stability of persistence modules*. SpringerBriefs in Mathematics. Springer, 2016.

Cohen-Steiner, D., Edelsbrunner, H., and Harer, J. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

Edelsbrunner, H. The union of balls and its dual shape. In *Proceedings of the ninth annual symposium on Computational geometry*, pp. 218–231, 1993.

Hofer, C., Graf, F., Rieck, B., Niethammer, M., and Kwitt, R. Graph filtration learning. In *37th International Conference on Machine Learning (ICML 2020)*, volume 119, pp. 4314–4323. PMLR, 2020.

Morris, C., Kriege, N., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. TUDataset: A collection of benchmark datasets for learning with graphs. In *CoRR*. arXiv:2007.08663, 2020.

Poulenard, A., Skraba, P., and Ovsjanikov, M. Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, volume 37, pp. 13–25. Wiley Online Library, 2018.

Wilkie, A. J. Model completeness results for expansions of the ordered field of real numbers by restricted pfaffian functions and the exponential function. *Journal of the American Mathematical Society*, 9(4):1051–1094, 1996.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations (ICLR 2019)*. OpenReviews.net, 2019.