REPRESENTATIVE DATASETS: THE PERCEPTRON CASE

A PREPRINT

Rocio Gonzalez-Diaz
Dept. of Applied Math I
University of Seville
rogodi@us.es

Miguel A. Gutiérrez-Naranjo
Dept. of Computer Sciences and
Artificial Intelligence
University of Seville
magutier@us.es

Eduardo Paluzo-Hidalgo
Dept. of Applied Math I
University of Seville
epaluzo@us.es

March 21, 2019

ABSTRACT

One of the main drawbacks of the practical use of neural networks is the long time needed in the training process. Such training process consists in an iterative change of parameters trying to minimize a loss function. These changes are driven by a dataset, which can be seen as a set of labeled points in an n-dimensional space. In this paper, we explore the concept of *representative dataset* which is smaller than the original dataset and satisfies a nearness condition independent of isometric transformations. The representativeness is measured using persistence diagrams due to its computational efficiency. We also prove that the accuracy of the learning process of a neural network on a representative dataset is comparable with the accuracy on the original dataset when the neural network architecture is a perceptron and the loss function is the mean squared error. These theoretical results accompanied with experimentation open a door to the size reduction of the dataset in order to gain time in the training process of any neural network.

Keywords Computational Learning Theory · Data Reduction · Perceptron · Neural Networks · Representative Datasets · Computational Topology.

1 Introduction

The success of the different architectures used in the framework of neural networks are doubtless [1]. Some of the recent achievements in areas such as generation of video images [2], recognition [3] or textual analysis [4] show the surprising potential of such architectures. In spite of such success, they still present some lacks. One of the main drawbacks of these systems is the long time needed in the training process. Such long training time is usually associated with two factors: first, the large amount of weights to be adjusted in the current architectures and, second, the huge datasets used to train neural networks. In general, the training time of a complex neural network from the scratch is so long that many current researches use *pretrained* neural networks as Oxford VGG models [5], Google Inception Model [6] or Microsoft ResNet Model [7]. Another attempts to reduce the training time are for example the followings. In [8], training time is reduced by partitioning the training task in multiple training subtasks with sub-models, which can be performed independently and in parallel. In [9], asynchronous averaged stochastic gradient descent is used. Lastly, a sampling-based approach to reduce data transmission is considered in [10].

Roughly speaking, such training process consists of a search of a local minimum of a loss function in an abstract space where the *states* are sets of weights. Each of the batch of training samples provides an extremely small change on the weights according to the training rules. The aim of such changes is to find the *best* set of weights which minimize the loss function. If we consider a *geometrical* interpretation of the learning process, such changes can be seen as tiny steps in a multidimensional metric space of parameters that follow the direction settled by the gradient vector of the loss function. In some sense, one can think that two *close* points of the dataset with the same label provide *similar information* to the learning process since the gradient vector of the loss function on such points is similar. Such viewpoint leads us to look for a *representative dataset*, “close" to and with a smaller number of points than the original dataset but keeping its “information", allowing the neural network to perform the learning process taking less time without losing accuracy.

In this paper, we reduce the training time by choosing a *representative* dataset of the original one. We formally prove that the accuracy of the neural network trained with the representative dataset is comparable to the accuracy of the neural network trained with the original dataset for the perceptron case. Besides, experimental evidence indicates the usefulness of representative datasets for the general case. Moreover, in order to “keep the shape" of the original dataset, the concept of representative dataset is associated with a notion of nearness independent of isometric transformations. Finally, as a first approach, Gromov-Hausdorff distance is used to measure this *representativeness*. Nonetheless, as Gromov-Hausdorff distance complexity is an open problem[1], bottleneck distance between persistence diagrams [12] is used instead as a lower bound to Gromov-Hausdorff distance since its complexity is cubic in the size of the dataset (see [13]). The implementation of the methodology can be found in the online supplementary material documentation[2].

The paper is organized as follows. In Section 2, basic definitions and results from artificial neural networks and persistent homology are given. The notion of representative dataset is introduced in Section 3. In Section 4, the perceptron architecture is studied to provide bounds to compare the training performances within the original and its representative dataset. Persistent homology is used in Section 5 to measure the representativeness of a dataset. In Section 6, experimental results are provided for the perceptron case pointing out the fact that it is equivalent to train a perceptron with a dataset or with its representative dataset. In Section 7, we illustrate the same fact through an experiment using a more complex neural network architecture. Finally, some conclusions and future work are provided in Section 8.

2 Background

Next we recall some basic definitions and notations used throughout the paper.

2.1 Neural Networks

The research field of neural networks is extremely vivid and new architectures are continuously being presented (see, e.g., CapsNets [7], or new variants of the *Gated Recurrent Units* [14, 15]), so the current definition of neural network is far from the classic definition of multilayer perceptron or radial basis function networks [16]. As a general setting, a neural network is a mapping $\mathcal{N}_{w,\Phi} : \mathbb{R}^n \to \mathbb{R}^m$ which depends on a set of weights w and a set of parameters Φ which involves the description of the synapses between neurons, layers, activation functions and whatever consideration in its architecture.

To train the neural network $\mathcal{N}_{w,\Phi}$, we use a *dataset* which is a finite set of pairs $\mathcal{D} = \{(x, c_x)\}$ where, for each pair (x, c_x), point x lies in $\mathbb{R}^n$ and label c_x lies in $[\![0, k]\!] = \{0, 1, \dots, k\}$, for some $k \in \mathbb{N}$.

To perform the learning process, we use: (1) a *loss function* which measures the difference between the output of the network (obtained with the current weights) and the desired output; and (2) a loss-driven *training* method to iteratively update the weights.

2.2 Persistent Homology

In this paper, the representativeness of a dataset will be measured using methods from the recent developed area called Computational Topology whose main tool is *persistent homology*. A detailed presentation of this field can be found in [12].

Homology provides mathematical formalism to study how a space is connected, being the m-dimensional homology group the mathematical representation for the m-dimensional holes in the space. This way, 0-dimensional homology group counts the connected components of the space, 1-dimensional homology group its tunnels and 2-dimensional homology group its cavities. For higher dimensions, the intuition of holes is lost.

Persistent homology is usually computed when exact homology can not be determined. An example of this casuistic can appear when a point cloud samples a surface. Persistent homology is based on the concept of *filtration*, which is an

[1]It seems to be intractable in practice [11].

[2]https://github.com/Cimagroup/Experiments-Representative-datasets

increasing sequence of simplicial complexes which change along time. The building blocks of a simplicial complex are m-simplices.

Definition 1. The m-simplex generated by $m+1$ affinely independent points $v^0, \dots, v^m \in \mathbb{R}^n$ (being $m \leq n$), denoted by $\mu = \langle v^0, \dots, v^m \rangle$, is defined by:

$$\mu = \left\{ x \in \mathbb{R}^n \;\middle|\; x = \sum_{i=0}^{m} \lambda_i v^i : \; \lambda_i \geq 0, \; \sum_{i=0}^{m} \lambda_i = 1 \right\}.$$

A face τ of μ is a simplex generated by any subset of $\{v^0, \dots, v^m\}$. In such case we denote $\tau \leq \mu$.

From the idea of m-simplex and face, the concepts of simplicial complex, subcomplex and filtration arise in a natural way.

Definition 2. A simplicial complex K is a finite set of simplices, satisfying the following properties:

1. If $\mu, \tilde{\mu} \in K$ and $\mu \cap \tilde{\mu} \neq \emptyset$, then $\mu \cap \tilde{\mu} \in K$.
2. If $\mu \in K$ and $\tau \leq \mu$, then $\tau \in K$.

A subset $\tilde{K}$ of K is a subcomplex of K if $\tilde{K}$ is also a simplicial complex.

A filtration of a simplicial complex K is a nested sequence of subcomplexes,

$$\emptyset = K_0 \subset K_1 \subset \cdots \subset K_n = K.$$

One example of filtration is the Vietoris-Rips filtration (see [17]). Roughly speaking, a Vietoris-Rips filtration is obtained by creating open balls from every point of a set in an n-dimensional space. The radius of the ball gets bigger along time and those vertices whose balls intersect are joined forming new simplices. Using this idea, the filtration is built. We say that a *hole is born* when it appears during the construction of the filtration and we say that a *hole dies* when it merges with another hole. The rule to decide which hole dies when they merge is the *elder rule* which establishes that younger holes die. So, births and deaths of all holes are controlled along time in the filtration. One of the common graphical representations of births vs. deaths along time is the so-called *persistence diagram* which is a set of points on the Cartesian plane. A point of a persistence diagram represents a hole in a concrete dimension. Since death happens only after births, all the points in a persistence diagram lie above the diagonal axis. Furthermore, those points in a persistence diagram that are far from the diagonal axis are candidates to be *topologically significant* since they represent holes which survive for a long time. The so-called bottleneck distance can be used to compare two persistence diagrams.

Definition 3. The Bottleneck distance between two persistence diagrams Dgm and $\widetilde{Dgm}$ is:

$$d_B(Dgm, \widetilde{Dgm}) = \inf_{\phi: Dgm \to \widetilde{Dgm}} \; \sup_{\alpha \in Dgm} ||\alpha - \phi(\alpha)||_\infty$$

where ϕ is any possible bijection between Dgm and $\widetilde{Dgm}$.

An useful result used in this paper is the following one that connects Gromov-Hausdorff distance between two metric spaces and bottleneck distance between the persistence diagrams of their corresponding Vietoris-Rips filtrations.

Theorem 1. *([18]) For any two subsets X and Y of $\mathbb{R}^n$, and for any dimension $q \leq n$, bottleneck distance between $Dgm_q(X)$ and $Dgm_q(Y)$ is bounded by Gromov-Hausdorff distance of X and Y:*

$$d_B(Dgm_q(X), Dgm_q(Y)) \leq 2 d_{GH}(X, Y).$$

Let us recall that Hausdorff distance between X and Y is:

$$d_H(X, Y) = \max\{ \sup_{x \in X} \inf_{y \in Y} ||x - y||, \sup_{y \in Y} \inf_{x \in X} ||x - y|| \},$$

and Gromov-Hausdorff distance between X and Y is defined as the infimum of the Hausdorff distance taken over all possible isometric transformations. That is,

$$d_{GH}(X, Y) = \inf_{f,g} \{ d_H\big(f(X), g(Y)\big) \},$$

where $f: X \to Z$ denotes an isometric transformation of X into some metric space Z and $g: Y \to Z$ an isometric transformation of Y into Z.

3 Representative Datasets

The key idea in this paper is the formal definition of representative datasets and the proof of their usefulness to decrease the time needed to train a neural network without loosing accuracy. Proving such general result for *all* the neural network architectures is out of the scope of this paper and, probably, it is not possible since the definition of *neural network* is continuously evolving over time as we mentioned before. In spite of such difficulties, we will start in this paper by showing the usefulness of representative datasets for the perceptron case.

In this section, we provide a definition of *representative datasets* which is general for all neural networks. The intuition behind this definition is to keep the *shape* of the original dataset while reducing its number of points.

Definition 4. A labeled point $(\tilde{x}, c_{\tilde{x}}) \in \mathbb{R}^n \times [\![0, k]\!]$ is ε-representative of $(x, c_x) \in \mathbb{R}^n \times [\![0, k]\!]$ if $c_x = c_{\tilde{x}}$ and there exists $\delta \in \mathbb{R}^n$ such that $x = \tilde{x} + \delta$ with $||\delta|| \leq \varepsilon$, where $\varepsilon \in \mathbb{R}$ is the representation error. We denote $\tilde{x} \approx_\varepsilon x$.

The next step is to define the concept of ε-representative dataset. Notice that if a dataset can be correctly classified by a neural network, any isometric transformation of such dataset is also correctly classified by the neural network. Therefore, the definition of ε-representative dataset should be independent of such transformations.

Definition 5. A dataset $\tilde{\mathcal{D}} = \{(\tilde{x}, c_{\tilde{x}}) : \tilde{x} \in \tilde{X} \subset \mathbb{R}^n$ and $c_{\tilde{x}} \in [\![0, k]\!]\}$ is ε-representative of $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^n$ and $c_x \in [\![0, k]\!]\}$ if there exists an isometric transformation $f : \tilde{X} \to \mathbb{R}^n$, such that for any $(x, c_x) \in \mathcal{D}$ there exists $(\tilde{x}, c_{\tilde{x}}) \in \tilde{\mathcal{D}}$ satisfying that $f(\tilde{x}) \approx_\varepsilon x$. We will say that ε is *optimal* if $\varepsilon = \min\{\xi : \tilde{\mathcal{D}}$ is a ξ-representative dataset of $\mathcal{D}\}$.

Proposition 1. Let $\tilde{\mathcal{D}}$ be a ε-representative dataset (with set of points $\tilde{X}$) of a dataset $\mathcal{D}$ (with set of points X). Then $d_{GH}(X, \tilde{X}) \leq \varepsilon$.

Proof. By definition of ε-representative dataset, there exists an isometric transformation from $\tilde{X}$ to $\mathbb{R}^n$ where for all $x \in X$ there exists $\tilde{x} \in \tilde{X}$ such that $||x - f(\tilde{x})|| \leq \varepsilon$. Therefore, $d_H(X, f(\tilde{X})) \leq \varepsilon$. Then, by definition of Gromov-Hausdorff distance,

$$d_{GH}(X, \tilde{X}) \leq d_H(X, f(\tilde{X})) \leq \varepsilon.$$

□

The definition of ε-representative datasets is not useful when ε is “big". The following result provides the optimal value for ε.

Remark 1. If $\varepsilon = d_{GH}(X, \tilde{X})$ then ε is optimal.

Therefore, one way to discern if a dataset $\tilde{\mathcal{D}}$ is “representative enough" of $\mathcal{D}$ is to compute Gromov-Hausdorff distance to X and $\tilde{X}$. If Gromov-Hausdorff distance is “big", we could say that the dataset $\tilde{\mathcal{D}}$ is not representative of $\mathcal{D}$. However, Gromov-Hausdorff distance is not useful in practice because of its high computational cost. An alternative approach to this problem is given in Section 5.

The following concept of λ-balanced dataset will be used in Section 4 to ensure the maintenance of similar results when training a perceptron with a representative dataset instead of the original one.

Definition 6. For $\lambda \in \mathbb{N}$, a dataset $\tilde{\mathcal{D}}$ is λ-balanced ε-representative of $\mathcal{D}$ if $\tilde{\mathcal{D}}$ is ε-representative of $\mathcal{D}$ and for each $(\tilde{x}, c_{\tilde{x}}) \in \tilde{D}$, $|\{(x, c_x) : f(x) \approx_\varepsilon \tilde{x}\}| = \lambda$ where f is a fixed isometric transformation.

Finally, we introduce the definition of *dominating set*, which will be used in the next subsection.

Definition 7. Given a graph $G = (X, E)$, a set $Y \subset X$ is a dominating set of X if for any $x \in X$, it is satisfied that $x \in Y$ or there exists $y \in Y$ adjacent to x.

3.1 Proximity Graph Algorithm

In this subsection, for a given ε, we propose a variant of the Proximity Graph Algorithm [19] to compute a ε-representative dataset $\tilde{\mathcal{D}}$ of a dataset $\mathcal{D} = \{(x, c_x) : x \in X$ and $c_x \in [\![0, k]\!]\}$. It works as follows. First, a proximity graph is built over X, establishing adjacency relations between points of X, represented by arcs. Second, using this graph, a dominating set $\tilde{X} \subseteq X$ is computed, obtaining a ε-representative dataset $\tilde{\mathcal{D}} = \{(\tilde{x}, c_{\tilde{x}}) : \tilde{x} \in \tilde{X}$ and $(\tilde{x}, c_{\tilde{x}}) \in \mathcal{D}\}$.

Definition 8. Given $X \subset \mathbb{R}^n$ and $\varepsilon \in \mathbb{R}$, an ε-proximity graph of X is a graph $G_\varepsilon(X) = (X, E)$ such that if $x, y \in X$ and $||x - y|| \leq \varepsilon$ then $(x, y) \in E$.

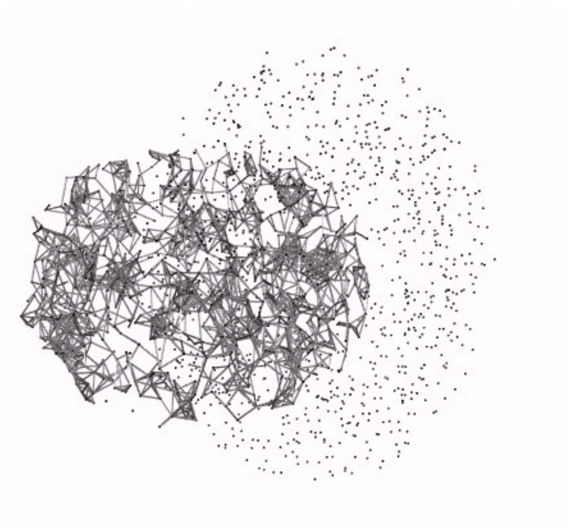

Figure 1: Point cloud sampling two interlaced solid torus and the proximity graph of one of them for a fixed ε.

See Fig. 1 in which the proximity graph of one of the two interlaced solid torus is drawn for a fixed ε.

The pseudo-code of the Proximity Graph Algorithm used in this paper is the following.

Algorithm 1: Proximity Graph Algorithm

```
Input: A dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^n$ and $c_x \in [\![0, k]\!]\}$.
Output: A dataset $\tilde{\mathcal{D}} \subset \mathcal{D}$.
for $c = 0$ to $c = k$ do
    $X_c = \{x : (x, c) \in \mathcal{D}$
    $\tilde{X}_c = \text{DominatingSet}(G_\varepsilon(X_c))$
$\tilde{X} = \bigcup_{c=0}^{k} \tilde{X}_c$
$\tilde{\mathcal{D}} = \{(x, c) : x \in \tilde{X}$ and $(x, c) \in \mathcal{D}\}$.
```

Here, DominatingSet$(G_\varepsilon(X_c))$ refers to a dominating set obtained from the proximity graph $G_\varepsilon(X_c)$. Among the existing algorithms in the literature to obtain a dominating set, we will use, in our experiments in Section 6, the algorithm proposed in [20] that runs in $O(|X| \cdot |E|)$. Therefore, the complexity of Algorithm 1 is $O(|X|^2 + |X| \cdot |E|)$ because of the size of the matrix of distances between points and the obtaining of the dominating set.

Lemma 1. The output, $\tilde{\mathcal{D}}$, of Algorithm 1 is a ε-representative dataset of $\mathcal{D}$.

Proof. Let us prove that for any $(x, c_x) \in \mathcal{D}$ there exists $(\tilde{x}, c_x) \in \tilde{\mathcal{D}}$ such that $x \approx_\varepsilon \tilde{x}$. Two possibilities can arise:

1. If $(x, c_x) \in \tilde{\mathcal{D}}$, it is done.
2. If $(x, c_x) \notin \tilde{\mathcal{D}}$, since $\tilde{X}_{c_x} \subset \tilde{X}$ is a dominating set of $G_\varepsilon(X_{c_x})$, then there exists $\tilde{x} \in \tilde{X}_{c_x}$ such that $\tilde{x}$ is adjacent to x in $G_\varepsilon(X_{c_x})$. Therefore, $(\tilde{x}, c_x) \in \tilde{\mathcal{D}}$ and $x \approx_\varepsilon \tilde{x}$.

□

4 The perceptron case

One of the simplest neural network architecture is the perceptron. Our goal in this section is to prove that training a perceptron with a representative dataset is equivalent, in terms of accuracy, to training it with the original dataset. Specifically, we provide results using *stochastic gradient descent* and the *gradient descent* algorithm. Notice that in the case of stochastic gradient descent, the number of updates needed to reach convergence usually increases with the size of data (see [1, Section 5.9]). We will restrict our interest to a binary classification problem. Therefore, our input is a dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^n \text{ and } c_x \in [\![0,1]\!]\}$.

A *perceptron*, $\mathcal{N}_w$, with weights $w = (w_0, w_1, \dots, w_n) \in \mathbb{R}^{n+1}$ is defined as:

$$\mathcal{N}_w(x) = \begin{cases} 1 & \text{if } y_w(x) \geq \frac{1}{2}, \\ 0 & \text{otherwise;} \end{cases}$$

being

$$\begin{aligned} y_w : \mathbb{R}^n &\to (0,1) \\ x &\mapsto \sigma(wx) \end{aligned}$$

where, for $x = (x_1, \dots, x_n) \in \mathbb{R}^n$, wx denotes the expression:

$$w_0 + w_1 x_1 + \cdots + w_n x_n,$$

and $\sigma : \mathbb{R} \to (0,1)$, defined as

$$\sigma(z) = \frac{1}{1+e^{-z}},$$

is the *sigmoid function*.

A useful property of the sigmoid function is the easy expression of its derivative.

Lemma 2. If $m \in \mathbb{N}$ and $z \in \mathbb{R}$ then

$$0 < (\sigma^m)'(z) = m\sigma^m(z)(1-\sigma(z)) \leq \left(\frac{m}{m+1}\right)^{m+1}.$$

Proof. First, $(\sigma^m)'(z) = m\sigma^m(z)(1-\sigma(z))$ is greater than 0 since $0 < \sigma(z) < 1$ for all $z \in \mathbb{R}$. Second, let us find the local extrema of $(\sigma^m)'$ by computing the roots of its derivative:

$$(\sigma^m)''(z) = m\sigma^m(z)(1-\sigma(z))(m-(m+1)\sigma(z)).$$

Now, $(\sigma^m)''(z) = 0$ if and only if $m - (m+1)\sigma(z) = 0$. The last expression vanishes at $z = \log(m)$. Besides, $(\sigma^m)''(z) > 0$ if and only if $m-(m+1)\sigma(z) > 0$ which is true for all $z \in (-\infty, \log(m))$. Analogously, $(\sigma^m)''(z) < 0$ for all $z \in (\log(m), +\infty)$, concluding that $z = \log(m)$ is a global maximum. Finally, $(\sigma^m)'(\log(m)) = \left(\frac{m}{m+1}\right)^{m+1}$. □

In the following lemma, we prove that the difference in the output of the function y_w^m evaluated at a point x and at its ε-representative point $\tilde{x}$ depends on the weights w and the parameter ε.

Lemma 3. Let $w \in \mathbb{R}^{n+1}$ and $x, \tilde{x} \in \mathbb{R}^n$ with $\tilde{x} \approx_\varepsilon x$. Then,

$$||y_w^m(\tilde{x}) - y_w^m(x)|| \leq \rho_m ||w||_* \varepsilon,$$

where

$$\rho_m = \rho_{(wx, w\tilde{x}, m)} = \begin{cases} (\sigma^m)'(z), & \text{if } \log(m) < z, \\ (\sigma^m)'(\tilde{z}), & \text{if } \tilde{z} < \log(m), \\ (\sigma^m)'(\log(m)), & \text{otherwise.} \end{cases}$$

with $z = \min\{wx, w\tilde{x}\}$ and $\tilde{z} = \max\{wx, w\tilde{x}\}$.

Proof. Let us assume, without loss of generality, that $wx \leq w\tilde{x}$. Then, using Mean Value Theorem, there exists $\beta \in (wx, w\tilde{x})$ such that

$$y_w^m(\tilde{x}) - y_w^m(x) = (\sigma^m)'(\beta)(w\tilde{x} - wx).$$

By Lemma 2, the maximum of $(\sigma^m)'$ in the interval $[z, \tilde{z}]$ is reached at $\log(m)$ if $z < \log(m) < \tilde{z}$, at z if $\log(m) \leq z$, and at $\tilde{z}$ if $\tilde{z} \leq \log(m)$, with $z = \min\{wx, w\tilde{x}\}$ and $\tilde{z} = \max\{wx, w\tilde{x}\}$. Consequently,

$$||y_w^m(x) - y_w^m(x)|| \leq \rho_{(wx, w, \tilde{x}, m)} ||w(\tilde{x} - x)||. \tag{1}$$

Applying now Hölder's inequality we obtain:

$$||w(\tilde{x} - x)|| \leq ||w||_* ||\tilde{x} - x|| \leq ||w||_* \varepsilon.$$

Replacing $||w(\tilde{x} - x)||$ by $||w||_* \varepsilon$ in Eq. (1), we obtain the desired result. □

The following is a direct consequence of Lemma 2 and Lemma 3.

Corollary 1. Let $w \in \mathbb{R}^{n+1}$ and $x, \tilde{x} \in \mathbb{R}^n$ with $\tilde{x} \approx_\varepsilon x$. Then,

$$||y_w^m(\tilde{x}) - y_w^m(x)|| \le \left(\frac{m}{m+1}\right)^{m+1} ||w||_* \varepsilon.$$

As previously anticipated, we will use stochastic gradient descent to train the perceptron, which tries to minimize the following error function:

$$\mathbb{E}(w, X) = \mathbb{E}(E_x(w)) = \frac{2}{|X|} \sum_{x \in X} E_x(w),$$

where, for $(x, c_x) \in \mathcal{D}$ and $w \in \mathbb{R}^{n+1}$,

$$E_x(w) = \frac{1}{2}(c_x - y_w(x))^2$$

is the loss function considered in this paper.

Following stochastic gradient descent, for all $j \in [\![0, n]\!]$, the weights $w^i = (w_0^i, w_1^i, \dots, w_n^i)$ of the ith-state of the perceptron is updated according to the following rule:

$$w_j^{i+1} \leftarrow w_j^i - \eta_i \frac{\partial E_{x^i}(w^i)}{\partial w_j^i}, \tag{2}$$

where $\eta_i > 0$ is the *learning rate*. Rule (2) takes a random point of the dataset for every iteration i, being able to repeat the same point in different iterations of the algorithm.

Let us notice that

$$\frac{\partial E_x(w)}{\partial w_j} = (c_x - y_w(x)) y'_w(x) x_j, \tag{3}$$

for $j \in [\![1, n]\!]$ and $x = (x_1, \dots, x_n)$. Besides,

$$\frac{\partial E_x(w)}{\partial w_0} = (c_x - y_w(x)) y'_w(x).$$

By abuse of notation we write Eq. (3) for all $j \in [\![0, n]\!]$, assuming that $x_0 = 1$.

Now, using Eq. (3), Rule (2) can be written as follows:

$$w^{i+1} \leftarrow w^i - \eta_i (c_i - y_{w^i}(x^i)) y_{w^i}(x^i)(1 - y_{w^i}(x^i)) x^i \tag{4}$$

for $(x^i, c_i) \in \mathcal{D}$.

To assure the convergence of the training process to a local minimum, it is enough that, for each iteration i, the learning rate η_i satisfies the following conditions (see [1, Section 8.3.1]):

$$\sum_{i=1}^{\infty} \eta_i = \infty \qquad \text{and} \qquad \sum_{i=1}^{\infty} \eta_i^2 < \infty$$

The following lemma determines when ε-representative points are classified under the same label as the points they represent.

Lemma 4. Let $\tilde{\mathcal{D}}$ be a ε-representative dataset of a dataset $\mathcal{D}$. Let $\mathcal{N}_w$ be a perceptron. Let $(x_0, c) \in \mathcal{D}$ and $(\tilde{x}_0, c) \in \tilde{\mathcal{D}}$ with $\tilde{x}_0 \approx_\varepsilon x_0$. If $\varepsilon \le \frac{||wx_0||}{||w||}$ then $\mathcal{N}_w(x_0) = \mathcal{N}_w(\tilde{x}_0)$.

Proof. First, if $wx = 0$ then $\varepsilon = 0$ and therefore $x_0 = \tilde{x}_0$. Let us then suppose, without loss of generality, that $wx < 0$. Then $y_w(x_0) < \frac{1}{2}$ and $\mathcal{N}_w(x_0) = 0$ by definition of perceptron. Now, since $\varepsilon < \frac{||wx_0||}{||w||}$ then x_0 and $\tilde{x}_0$ belong to the same semispace in which the space is divided by the hiperplane $wx = 0$. Therefore, $w\tilde{x}_0 < 0$, then $y_w(\tilde{x}) < \frac{1}{2}$ and finally $\mathcal{N}_w(\tilde{x}) = 0$. □

By Lemma 4, we can state that if ε is "small" enough, then $\mathcal{D}$ and $\tilde{\mathcal{D}}$ will produce the same accuracy on the perceptron $\mathcal{N}_w$. Let us define

$$\mathcal{I}_w(x) = \begin{cases} 1 & \text{if } c_x = \mathcal{N}_w(x), \\ 0 & \text{otherwise,} \end{cases}$$

for any $(x, c_x) \in \mathcal{D}$. The following result holds.

Theorem 2. *Let $\tilde{\mathcal{D}}$ be a λ-balanced ε-representative dataset of a dataset $\mathcal{D}$. Let $\mathcal{N}_w$ be a perceptron. If $\varepsilon \leq \min\left\{\frac{||wx||}{||w||} : (x, c_x) \in \mathcal{D}\right\}$ then*

$$\frac{1}{|\mathcal{D}|}\sum_{(x,c_x)\in\mathcal{D}} I_w(x) = \frac{1}{|\tilde{\mathcal{D}}|}\sum_{(\tilde{x},c_{\tilde{x}})\in\tilde{\mathcal{D}}} I_w(\tilde{x}).$$

Proof. Since $\tilde{\mathcal{D}}$ is λ-balance ε-representative of $\mathcal{D}$, then $|\mathcal{D}| = \lambda \cdot |\tilde{\mathcal{D}}|$ and we have:

$$\begin{aligned}
&\frac{1}{|\mathcal{D}|}\sum_{(x,c_x)\in\mathcal{D}} I_w(x) - \frac{1}{|\tilde{\mathcal{D}}|}\sum_{(\tilde{x},c_{\tilde{x}})\in\tilde{\mathcal{D}}} I_w(\tilde{x}) = \\
&= \frac{1}{|\mathcal{D}|}\sum_{(x,c_x)\in\mathcal{D}} \left(I_w(x) - \lambda \cdot I_w(\tilde{x})\right) \\
&= \frac{1}{|\mathcal{D}|}\sum_{(\tilde{x},c_{\tilde{x}})\in\tilde{\mathcal{D}}} \sum_{x\approx_\varepsilon \tilde{x}} \left(I_w(x) - I_w(\tilde{x})\right).
\end{aligned} \tag{5}$$

Finally, $I_w(x) = I_w(\tilde{x})$ for all $x \approx_\varepsilon \tilde{x}$ and $(\tilde{x}, c_{\tilde{x}}) \in \tilde{\mathcal{D}}$ by Lemma 4 since $\varepsilon < \frac{||wx||}{||w||}$ for all $(x, c_x) \in \mathcal{D}$. □

Next, we compare the two errors $\mathbb{E}(w, X)$ and $\mathbb{E}(w, \tilde{X})$ obtained when taking a dataset $\mathcal{D}$ and its λ-balanced ε-representative dataset $\tilde{\mathcal{D}}$.

Lemma 5. Let $X, \tilde{X} \subset \mathbb{R}^n$. Let $w \in \mathbb{R}^{n+1}$. Let $\mathcal{D} = \{(x, c_x) : x \in X \text{ and } c_x \in [\![0, 1]\!]\}$ and $\tilde{\mathcal{D}} = \{(\tilde{x}, c_{\tilde{x}}) : \tilde{x} \in \tilde{X} \text{ and } c_{\tilde{x}} \in [\![0, 1]\!]\}$. Suppose $\tilde{\mathcal{D}}$ is λ-balanced ε-representative of $\mathcal{D}$. Then:

$$\big|\big|\mathbb{E}(w, X) - \mathbb{E}(w, \tilde{X})\big|\big| \leq \frac{1}{|X|}\sum_{x\in X}\big(2c_x\rho_1 + \rho_2\big)||w(x - \tilde{x})||$$

where $\rho_m = \rho_{(wx, w\tilde{x}, m)}$ for $m = 1, 2$, and for each addend, $x \approx_\varepsilon \tilde{x}$.

Proof.

$$\begin{aligned}
\mathbb{E}(w, X) - \mathbb{E}(w, \tilde{X}) =& \frac{1}{|X|}\sum_{x\in X}(c_x - y_w(x))^2 \\
&- \frac{1}{|\tilde{X}|}\sum_{\tilde{x}\in\tilde{X}}(c_{\tilde{x}} - y_w(\tilde{x}))^2 \\
=& \frac{1}{|X|\cdot|\tilde{X}|}\Big(|\tilde{X}|\sum_{x\in X}(c_x - y_w(x))^2 \\
&\qquad - |X|\sum_{\tilde{x}\in\tilde{X}}(c_{\tilde{x}} - y_w(\tilde{x}))^2\Big).
\end{aligned}$$

On both addends of the last expression we have the same number of elements. Since $\tilde{D}$ is λ-balanced ε-representative of $\mathcal{D}$ then $|X| = \lambda\,|\tilde{X}|$. Then,

$$\begin{aligned}
&||\mathbb{E}(w, X) - \mathbb{E}(w, \tilde{X})|| \\
&= \frac{1}{|X|}\Big|\Big|\sum_{x\in X} 2c_x\big(y_{\tilde{w}}(\tilde{x}) - y_w(x)\big) + y_w^2(x) - y_{\tilde{w}}^2(\tilde{x})\Big)\Big|\Big| \\
&\leq \frac{1}{|X|}\sum_{x\in X} 2c_x||y_{\tilde{w}}(\tilde{x}) - y_w(x)|| + ||y_w^2(x) - y_{\tilde{w}}^2(\tilde{x})||,
\end{aligned}$$

where, for each addend, $\tilde{x} \approx_\varepsilon x$. Applying Lemma 3 for $m = 1, 2$ to the last expression, we get:

$$||\mathbb{E}(w, X) - \mathbb{E}(w, \tilde{X})|| \leq \frac{1}{|X|}\sum_{x\in X}\big(2c_x\rho_1 + \rho_2\big)||w(x - \tilde{x})||.$$

□

From this last result we can infer the following. We can always find ε “small enough" so that the difference between the errors obtained from one dataset and from another which is ε-representative of the former is “close" to zero.

Theorem 3. *Let $\delta > 0$. Let $\tilde{\mathcal{D}}$ be a λ-balanced ε-representative dataset of a dataset $\mathcal{D}$. Let $\mathcal{N}_w$ be a perceptron. If $\varepsilon \leq \frac{54}{43||w||_*}\delta$, then $||\mathbb{E}(w, X) - \mathbb{E}(w, \tilde{X})|| < \delta$.*

Proof. First, $\rho_1 \leq \frac{1}{4}$ and $\rho_2 \leq \frac{8}{27}$ by Corollary 1. Second, since $c_x \in [\![0, 1]\!]$, then we have:

$$\begin{aligned}
&\frac{1}{|X|}\sum_{x\in X}\big(2c_x\rho_1 + \rho_2\big)||w(x - \tilde{x})|| \\
&\leq \frac{1}{|X|}\sum_{x\in X}\Big(\frac{1}{2} + \frac{8}{27}\Big)||w(x - \tilde{x})|| \\
&= \frac{43}{54}\frac{1}{|X|}\sum_{x\in X}||w(x - \tilde{x})||.
\end{aligned}$$

Applying Hölder inequality to the last expression we get:

$$\frac{1}{|X|}\sum_{x\in X}\big(2c_x\rho_1 + \rho_2\big)||w(x - \tilde{x})|| \leq \frac{43}{54}||w||_*\varepsilon.$$

Therefore, by Lemma 5, if $\varepsilon \leq \frac{54}{43||w||_*}\delta$, then $||\mathbb{E}(w, X) - \mathbb{E}(w, \tilde{X})|| < \delta$ as stated. □

Summing up, in the case of stochastic gradient descent we have proved that it is equivalent to train a perceptron with a dataset $\mathcal{D}$ or with its λ-balanced ε-representative dataset. This fact will be highlighted in Section 6 for the perceptron case and in Section 7 for a neural network with a more complex architecture.

5 On the Application of Persistent Homology

In this section, we recall the roll of persistent homology as a tool to infer the representativeness of a dataset.

First, from Theorem 1 in page 3, we can establish that bottleneck distance between persistence diagrams is a lower bound of the representativeness of the dataset.

Theorem 4. *Let $\tilde{\mathcal{D}}$ be a ε-representative dataset (with set of points $\tilde{X}$) of a dataset $\mathcal{D}$ (with set of points X). Let $dgm(X)$ and $dgm(\tilde{X})$ be the persistence diagrams of X and $\tilde{X}$, respectively. Then,*

$$d_B\big(dgm(X), dgm(\tilde{X})\big) \leq 2\varepsilon.$$

Proof. Since $\tilde{\mathcal{D}}$ is ε-representative of $\mathcal{D}$ then $d_{GH}(X, \tilde{X}) \leq \varepsilon$ by Proposition 1. Now, by Theorem 1,

$$\frac{1}{2}d_B(Dgm_p(X), Dgm_p(Y)) \leq d_{GH}(X, Y) \leq \varepsilon.$$

□

As a direct consequence of Theorem 4 and the fact that Hausdorff distance is an upper bound of Gromov-Hausdorff distance we have the following.

Corollary 2. Let $\tilde{\mathcal{D}}$ be a ε-representative dataset of $\mathcal{D}$ with optimal ε. Let $dgm(X)$ and $dgm(\tilde{X})$ be the persistence diagrams of X and $\tilde{X}$, respectively. Then,

$$\frac{1}{2}d_B\big(Dgm_p(X), Dgm_p(\tilde{X}))\big) \leq \varepsilon \leq d_H\big(X, \tilde{X}\big).$$

To illustrate the usefulness of this last result, we will discuss a simple example. In Fig. 2, a binary classification problem is given: Two classes corresponding, respectively, to the upper and the lower semicircumference together with a subsample of the circumference are shown. In Fig. 3, we can see a subset of the original dataset given in Fig. 2 and a possible decision boundary. As we can see, these two datasets do not represent the same classification problem. The decision boundaries are different and we should not consider the datasets to represent the same classification

Dataset	b_0	b_1	Hausdorff distance
Representative dataset 1	0.5	0.6	0.35
Representative dataset 2	0.85	0.68	0.94

Table 1: 0-dimensional and 1-dimensional Botleneck distances and Hausdorff distance between the datasets given in Fig. 4 and Fig. 3 with the dataset given in Fig. 2.

problem. However, the subset shown in Fig. 4 is more accurate, keeping the representativeness of the original dataset. Besides, we could say that minimizing the error function in the case of Fig. 4 is almost the same as minimizing the error function in the case of Fig. 2. This, that is evident for the eye, can be determined using bottleneck distance between the corresponding persistence diagrams and Hausdorff distance as follows. Compute the bottleneck distance between the two persistence diagrams and the Hausdorff distance between the two datasets to obtain the values shown in Table 1.

Using Corollary 2, we can infer that $0.3 \leq \varepsilon \leq 0.35$ for the dataset given in Fig. 4 and $0.425 \leq \varepsilon \leq 0.94$ for the dataset given in 3. Therefore, the dataset given in Fig. 4 can be considered more representative of the dataset showed in Fig. 2 than the dataset pictured in Fig. 2, as expected.

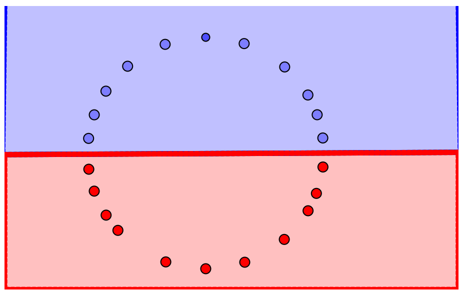

Figure 2: A binary classification problem given by a sampled circumference. In this case, the classification problem tries to distinguish between the upper and the lower semicircumference.

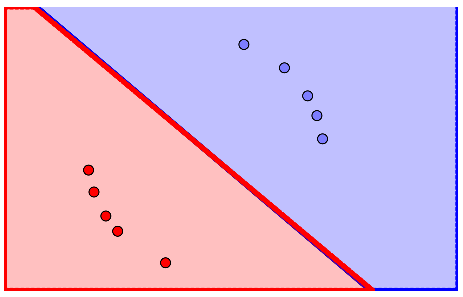

Figure 3: **(Representative dataset 2)** A subset of the binary classification problem given in Fig. 2. As we can see, it is not really representative of the original dataset.

6 Experimental results: The perceptron case

In this section, two experiments ((1) and (2)) are provided for the perceptron case, to support our results and illustrate the usefulness of our method. In experiment (1), two interlaced solid torus are sampled and, in (2), the Iris dataset is considered. In both experiments, the perceptron is initiated with the same weight and trained with three different datasets: the original dataset, a dominating dataset (which is the output of Algorithm 1) and a random dataset of the same size as the dominating dataset. These experiments support that a perceptron trained with representative datasets give similar accuracy than a perceptron trained with the original dataset. Besides, we show that the training time, in the case of the gradient descent training, is lower when using a representative dataset, and the convergence, in the case of stochastic gradient descent, is, generally, faster. Furthermore, computed bottleneck and Hausdorff distance are, respectively, lower and upper bounds of the Gromov-Hausdorff distance which is our estimator of representativeness.

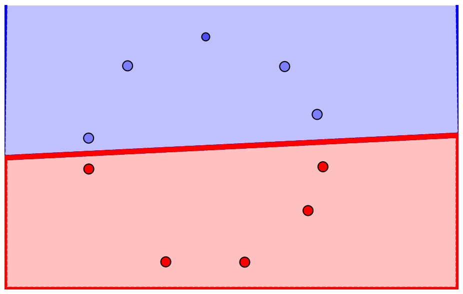

Figure 4: **(Representative dataset 1)** A subset of the binary classification problem given in Fig. 2. In this case, we can say that it is representative of the original dataset.

Datasets		Size	Hausdorff distance	Bottleneck distance
Interlaced torus	Original	4000		
	Dominating	267	0.69	0.41
	Random	267	1.03	0.44
Iris	Original	100		
	Dominating	20	0.56	0.31
	Random	20	0.9	0.81
Digits	Original	1797		
	Dominating	195	0.59	0.17
	Random	195	0.71	0.17

Table 2: Hausdorff distance and 0-dimensional bottleneck distance between the original dataset and the dominating or random dataset. For every problem (interlaced torus, iris and digits) we have three different datasets: original, dominating and random.

6.1 Two interlaced solid torus dataset

This experiment consists in a binary classification problem where each torus corresponds to a class. Algorithm 1 was applied to obtain a representative dataset call *dominating dataset* smaller than the original one. A random dataset with the same number of points as the dominating dataset was also computed. In Fig. 7, Fig. 8 and Fig. 9, the three datasets are shown: the original dataset, the dominating dataset and the random dataset. In Fig. 19, Fig. 20 and Fig. 21, the persistence diagrams of these datasets are shown and, in Table 2, Hausdorff distances and 0-dimensional bottleneck distances are provided. We can observe that the bottleneck distance and the Hausdorff distance are higher for the random dataset than for the dominating dataset when comparing them with the original dataset.

Perceptrons, initiated with same weights, were trained with stochastic gradient descent using the different datasets. They were trained along 100 iterations. Table 3 illustrates that the training time improves when using smaller datasets.

Fig. 5 shows the accuracy of the perceptron, trained with the different datasets using stochastic gradient descent, measured on the original dataset, illustrating that the training process were successful for all of them. Nevertheless, in Fig. 6, we can see the accuracy of the trained perceptrons measured on the original dataset. All of them converge to similar accuracy. We conjecture that the original dataset considered is too uniform to get different accuracy. However, we can see that the dominating dataset has a faster convergence than the other two datasets.

In Table 4, the means of the error differences obtained after the different training processes are shown. The maximum and minimum error value were 0.38 and 0.18, respectively, for the random dataset, and 0.19 and 0.18 for the original and the dominating dataset. Therefore, we can assert that the dominating dataset generalizes better the original dataset than the random one.

6.2 Iris dataset

In this experiment we used the Iris Dataset[3] which corresponds to a classification problem with three classes. It is composed by 150 4-dimensional instances. We limited our experiment to two of the three classes, keeping a balanced dataset of 100 points. Algorithm 1 was applied to obtain an ε-representative dataset of 20 points with $\varepsilon \leq 0.5$ (called

[3]https://archive.ics.uci.edu/ml/datasets/iris

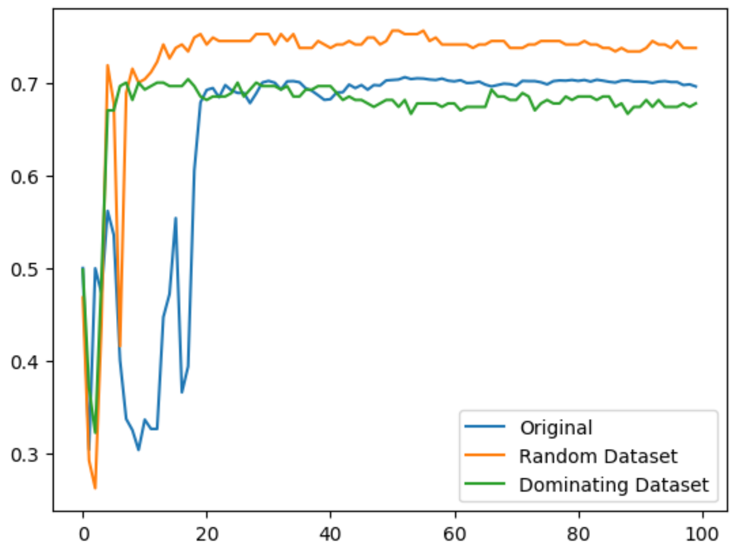

Figure 5: **(Two interlaced solid torus dataset)** Accuracy of the perceptron, trained with the different datasets using stochastic gradient descent, measured on the datasets that were used for its training process. It shows that all the training processes were successful. X-axis corresponds to the number of epochs and Y-axis to the accuracy.

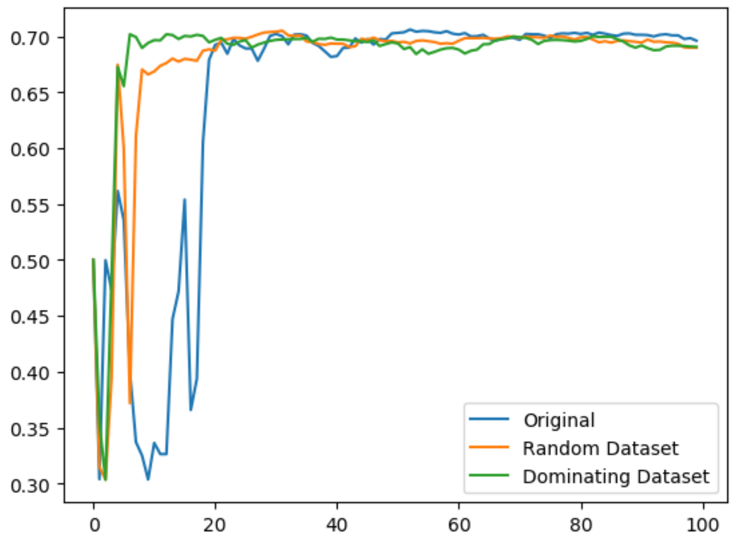

Figure 6: **(Two interlaced solid torus dataset)** Accuracy of the perceptron, trained with the different datasets using stochastic gradient descent, measured on the original dataset. X-axis corresponds to the number of epochs and Y-axis to the accuracy.

dominating dataset). A random dataset extracted from the original one with the same number of points than the dominating dataset was also computed for testing. These datasets can be seen in a three-dimensional representation in Fig. 10, Fig. 11 and Fig. 12, respectively. Besides, the associated persistence diagrams are shown in Fig. 22, 23 and 24. Hausdorff distance and 0-dimensional bottleneck distance between the original dataset and the dominating and random datasets are given in Table 2.

In Fig. 16 a training sample is shown. We can observe that the training process was successful for both the original and the dominating dataset. However, the random dataset got stagnant. Besides, Fig. 17 shows that the convergence was faster for the dominating dataset than for the other datasets and that the random dataset can not reach an accuracy over the original dataset comparable to the other two. We trained the perceptron with different initial weights and observed that the perceptron trained with the dominating dataset converges to an error similar to that of the perceptron trained

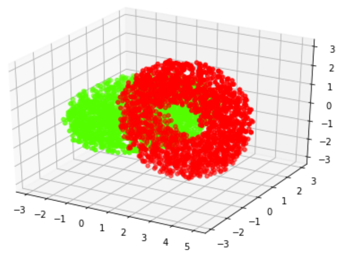
Figure 7: Toroids dataset

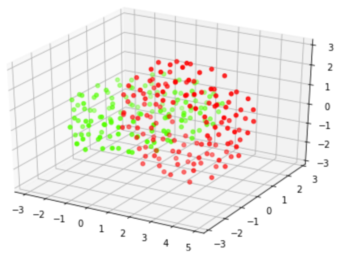
Figure 8: Dominating dataset toroids

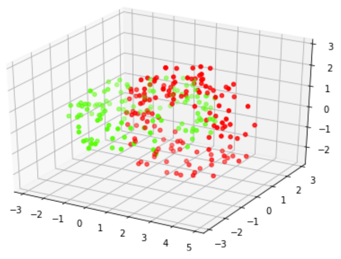
Figure 9: Random dataset toroid

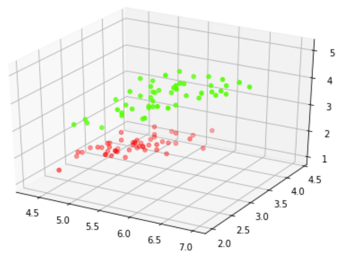
Figure 10: Iris dataset

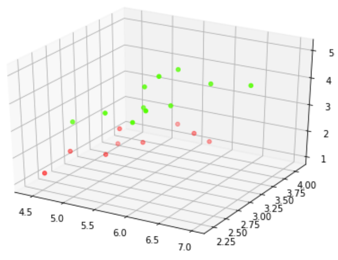
Figure 11: Dominating dataset Iris

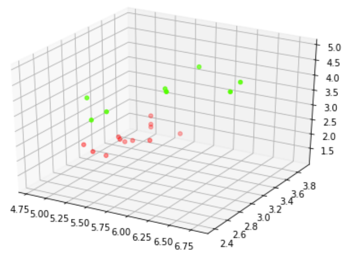
Figure 12: Random dataset Iris

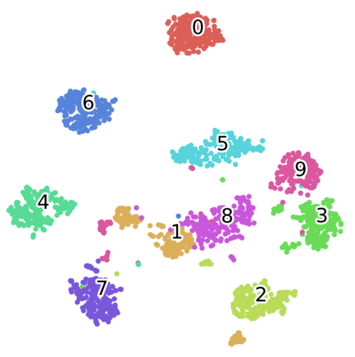
Figure 13: T-SNE embedding digits

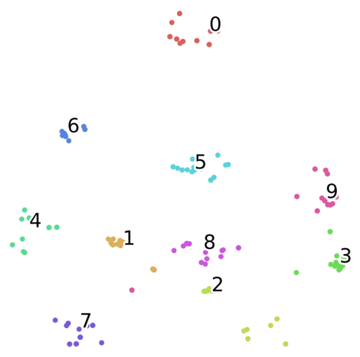
Figure 14: T-SNE embedding dominating dataset digits

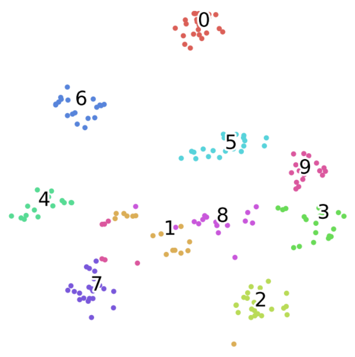
Figure 15: T-SNE embedding random dataset digits

with the original dataset. However, the random dataset seems to present a different error surface and the perceptron trained with such dataset can not achieve an error similar to that of the perceptron trained with the original dataset. This can be appreciated in Table 4, where the error differences and the bounds of Lemma 5 for the random and the dominating dataset are shown. The maximum and minimum error value were 0.71 and 0.004, respectively, for the random dataset, 0.45 and 0.004 for the dominating dataset and 0.5 and 0.003 for the original dataset. The previously given values and the values given in Table 4 were obtained after training the perceptron 100 times using different weights. Therefore, we can assert that the dominating dataset generalizes better the original dataset than the random one.

7 Experimental results: Digits dataset

The following experiment is quite different from the above. The dataset[4] we will use consists of images of digits classified in 10 different classes. This way, the neural network to train should classify images of digits into its corresponding digit from 0 to 9. An example of an image of each class can be seen in Fig. 18. The dataset is composed by 1797 64-dimensional instances. Algorithm 1 was applied to obtain a dominating dataset of size 195. In Fig. 13, Fig.14 and Fig.15 a two-dimensional projection of these datasets obtained using T-SNE (see [21]) are shown. Besides, the corresponding persistence diagrams can be seen in Fig. 25, fig. 26 and27. Hausdorff and bottleneck distances are

[4]`https://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html`

Datasets		Time (s)	
		Alg. 1	Training
Interlaced torus	Original	39.21	
	Dominating	1.43	5.84
Iris	Original	4.08	
	Dominating	0.07	1.28
Digits	Original	1147.32	
	Dominating	0.78	358.51

Table 3: Time in seconds for computing the dominating dataset (Algorithm 1) and for the training process over the different datasets we are dealing with in the paper. The training method consists in the gradient descent algorithm. In the case of the Digits dataset, 100000 epochs were needed to reach high accuracy since the architecture used was quite simple.

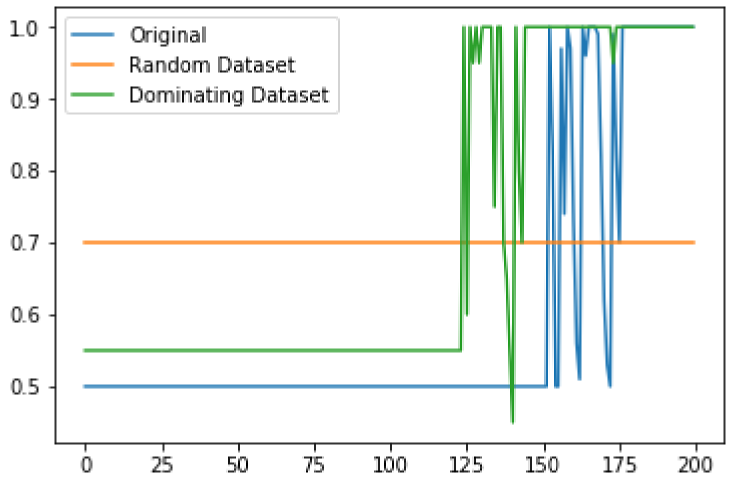

Figure 16: **(Iris dataset)** Accuracy of the perceptron, trained with the different datasets using stochastic gradient descent, measured on the datasets that were used for its training process. It shows that all the training processes were successful. X-axis corresponds to the number of epochs and Y-axis to the accuracy.

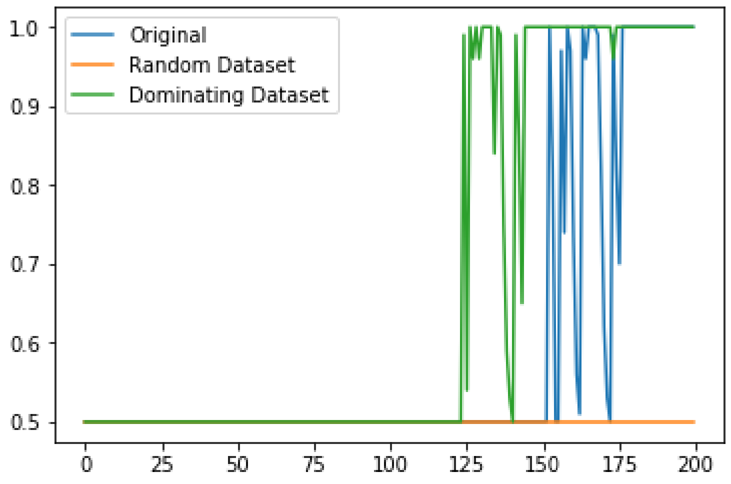

Figure 17: **(Iris dataset)** Accuracy of the perceptron, trained with the different datasets using stochastic gradient descent, measured on the original dataset. The line of the random dataset diverge from the rest. It shows that the dominating dataset generalizes better the original dataset than the random one. X-axis corresponds to the number of epochs and Y-axis to the accuracy.

Datasets		$\|\|E(w,X) - E(w,\tilde{X})\|\|$	Bound
Interlaced torus	Dominating	0.008	0.35
	Random	0.13	0.48
Iris	Dominating	0.05	0.14
	Random	0.4	0.52

Table 4: Comparison of the exact error differences and the bound given in Lemma 5 computed over the random and the dominating dataset, respectively, for the interlaced tours and the iris classification problem. The values correspond to the mean of the exact error differences and the bound obtained using 100 different random weights. As the datasets were not λ-balanced, an approximation of the bound is given, letting us to increase the multiplicity of some points in the computation.

shown in Table 2. In this case we used a neural network with $64 \times 32 \times 10$ neurons with sigmoid activation function in the hidden layer and softmax activation function in the output layer. The neural network was trained using stochastic gradient descent and categorical cross entropy as loss function. It was launched 100 times for the dominating dataset and a random dataset of the same size. The accuracy measured on the dominating dataset reached after training the neural network using the dominating dataset was 0.82. Similarly, the accuracy measured on the original dataset was 0.89. Therefore, it seems that the dominating dataset generalizes well the original dataset. The accuracy measured on the random dataset reached after training the neural network using the random dataset was 0.92. Similarly, the accuracy measured on the original dataset was 0.87. It seems that the model had overfitting with the random dataset and the performance over the original dataset is lower than over itself. Besides, the accuracy over the original dataset is higher for the dominating dataset. Then, the dominating dataset outperforms the random dataset.

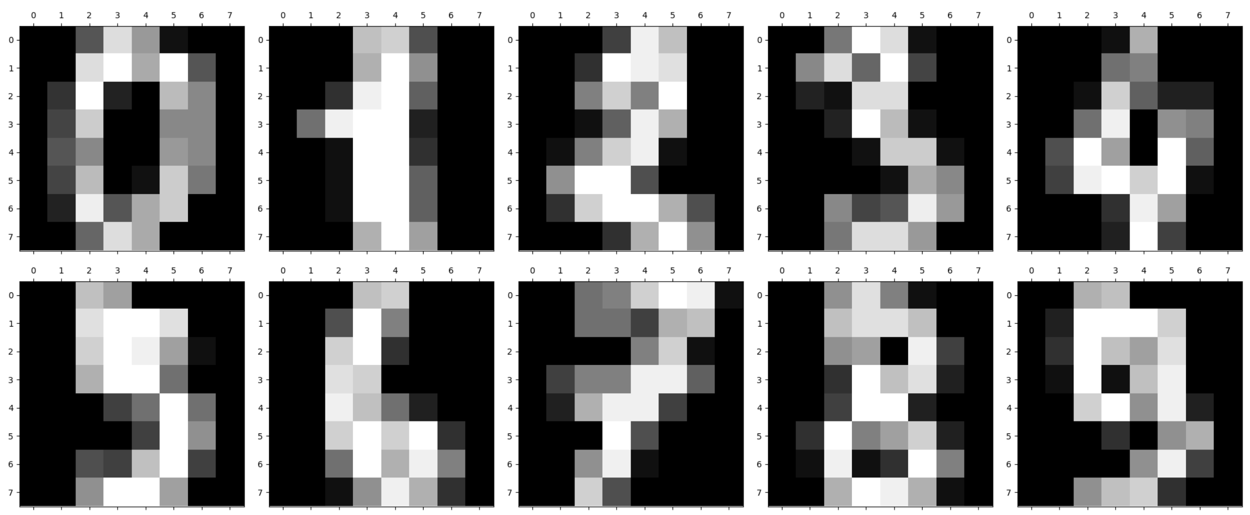

Figure 18: **(Digits dataset)** Example of an image of each class of the digits dataset. They are 32×32 arrays in gray scale.

8 Conclusions and future work

The success of practical applications and the availability of new hardware (e.g., GPUs [22] and TPUs [23]) have led to focus the neural networks research on the development of new architectures rather than on theoretical issues. Such new architectures are one of the pillars of the future research in neural networks, but a deeper understanding of the data structure is also necessary in order to the development of the field, as, for example, new discoveries on adversarial examples [24] have shown, or the one given in [25], where it is empirically shown the redundancy of several datasets.

In this paper, we propose the use of *representative datasets* as a new approach to reduce the learning time in neural networks based on the topological structure of the dataset. We have defined representative datasets using a notion of nearness that has Gromov-Hausdorff distance as lower bound. Nevertheles, bottleneck distance of persistence diagrams (which is a lower bound of Gromov-Hausdorff distance) is used to measure representativeness since it is computationally less expensive. Furthermore, the agreement between the provided theoretical results and the experiments supports that

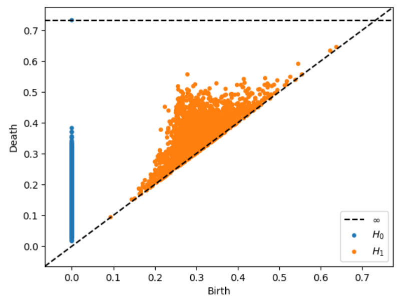

Figure 19: **(Two interlaced solid torus dataset)** Persistence diagram of the dataset given in Fig. 7

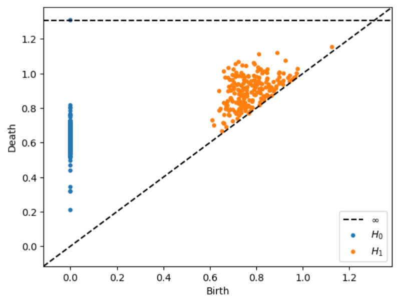

Figure 20: **(Two interlaced solid torus dataset)** Persistence diagram of the dominating dataset given in Fig 8.

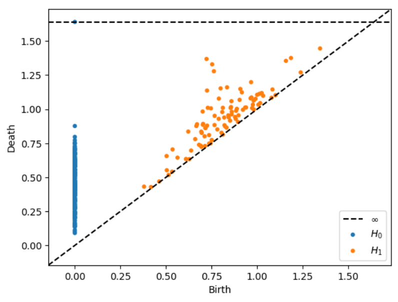

Figure 21: **(Two interlaced solid torus dataset)** Persistence diagram of the random dataset given in Fig. 9.

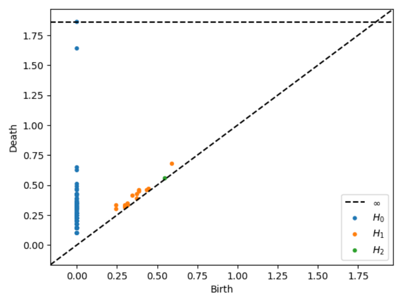

Figure 22: **(Iris dataset)** Persistence diagram of the dataset given in Fig. 10.

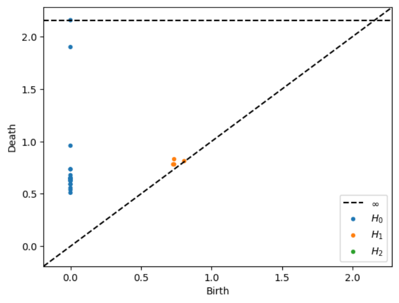

Figure 23: **(Iris dataset)** Persistence diagram of the dataset given in Fig. 11.

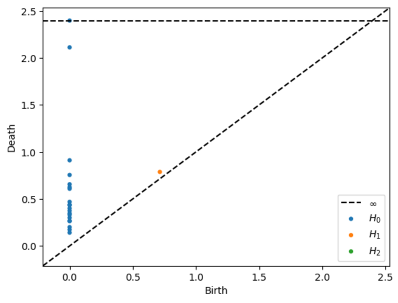

Figure 24: **(Iris dataset)** Persistence diagram of the random dataset given in Fig. 12.

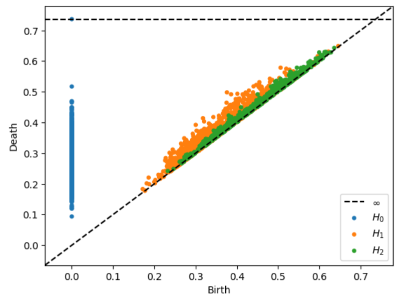

Figure 25: **(Digits dataset)** Persistence diagram of the dataset given in Fig. 13.

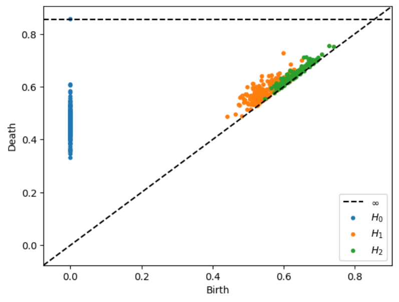

Figure 26: **(Digits dataset)** Persistence diagram of the dominating dataset given in Fig. 14.

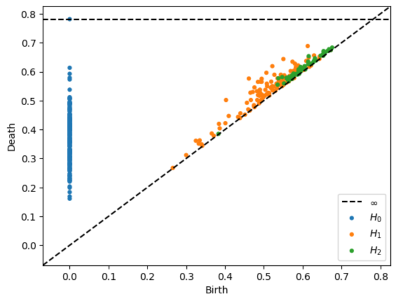

Figure 27: **(Digits dataset)** Persistence diagram of the random dataset given in Fig. 15.

representative datasets can be a good approach to reach an efficient summarization of a dataset to train a neural network. First, in the case of gradient descent, the training process is significantly faster and, second, in the case of stochastic gradient descent the convergence is usually faster, when using a representative dataset.

Planned future work is to provide more experiments using high dimensional real data and different reduction algorithms. Furthermore, we plan to formally prove that the proposed approach could be extended to different neural network architectures and training algorithms.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Nießner, Patrick Pérez, Christian Richardt, Michael Zollöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on*

Graphics (TOG), 37(4), 2018.

[3] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):677–691, 2017.

[4] Alexandra N. M. Darmon, Marya Bazzi, Sam D. Howison, and Mason A. Porter. Pull out all the stops: Textual analysis via punctuation sequences, 2018.

[5] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[8] Conrado S. Miranda and Fernando J. Von Zuben. Reducing the training time of neural networks by partitioning, 2015.

[9] Zhao You and Bo Xu. Improving training time of deep neural networkwith asynchronous averaged stochastic gradient descent. In *ISCSLP*, pages 446–449. IEEE, 2014.

[10] Tong Xiao, Jingbo Zhu, Tongran Liu, and Chunliang Zhang. Fast parallel training of neural language models. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4193–4199, 2017.

[11] Felix Schmiedl. Computational aspects of the gromov-hausdorff distance and its application in non-rigid shape matching. *Discrete & Computational Geometry*, 57(4):854–880, 2017.

[12] Herbert Edelsbrunner and John L. Harer. *Computational Topology, An Introduction*. American Mathematical Society, 2010.

[13] F. Chazal, D. Cohen-Steiner, L. J. Guibas, F. Mémoli, and S. Y. Oudot. Gromov-Hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum (proc. SGP 2009)*, pages 1393–1403, 2009.

[14] Rahul Dey and Fathi M. Salem. Gate-variants of gated recurrent unit (gru) neural networks, 2017.

[15] Joel Heck and Fathi M. Salem. Simplified minimal gated unit variations for recurrent neural networks, 2017.

[16] Simon S. Haykin. *Neural networks and learning machines*. Pearson Education, Upper Saddle River, NJ, third edition, 2009.

[17] J.C. Hausmann. *On the Vietoris-Rips Complexes and a Cohomology Theory for Metric Spaces*. Publications internes de la Section de mathématiques de l'Université de Genève. Université de Genève-Section de mathématiques, 1994.

[18] Frédéric Chazal, Vin de Silva, and Steve Oudot. Persistence stability for geometric complexes. *Geometriae Dedicata*, 173(1):193–214, Dec 2014.

[19] Rocio Gonzalez-Diaz, Miguel A. Gutiérrez-Naranjo, and Eduardo Paluzo-Hidalgo. Representative datasets for neural network. *Electronic Notes in Discrete Mathematics*, 68C:89–94, 2018.

[20] David W. Matula. Determining edge connectivity in o(nm). In *FOCS*, pages 249–251. IEEE Computer Society, 1987.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[22] Jiazhen Gu, Huan Liu, Yangfan Zhou, and Xin Wang. Deepprof: Performance analysis for deep learning applications via mining gpu execution patterns, 2017.

[23] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit, 2017.

[24] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning, 2017.

[25] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. In *ICLR*, 2019.