Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07252-y

ORIGINAL ARTICLE

Topology-based representative datasets to reduce neural network training resources

Rocio Gonzalez-Diaz[1] · Miguel A. Gutiérrez-Naranjo[2] · Eduardo Paluzo-Hidalgo[1]

Received: 5 October 2021 / Accepted: 29 March 2022

Abstract
One of the main drawbacks of the practical use of neural networks is the long time required in the training process. Such a training process consists of 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 a *representative dataset* which is a dataset smaller than the original one, satisfying a nearness condition independent of isometric transformations. Representativeness is measured using persistence diagrams (a computational topology tool) due to its computational efficiency. We theoretically prove that the accuracy of a perceptron evaluated on the original dataset coincides with the accuracy of the neural network evaluated on the representative dataset when the neural network architecture is a perceptron, the loss function is the mean squared error, and certain conditions on the representativeness of the dataset are imposed. These theoretical results accompanied by experimentation open a door to reducing the size of the dataset to gain time in the training process of any neural network.

Keywords Data reduction · Neural networks · Representative datasets · Computational topology

1 Introduction

The success of the different architectures used in the framework of neural networks is doubtless [1]. The achievements made in areas such as video imaging [2], recognition [3], or language models [4] show the surprising potential of such architectures. In spite of such success, they still have some shortcomings. One of their main drawbacks is the long time needed in the training process. Such a 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 time needed to train a complex neural network from the scratch is so long that many researchers use *pretrained* neural networks as, for example, Oxford VGG models [5], Google Inception Model [6], or Microsoft ResNet Model [7]. Other attempts to reduce the training time are, for example, to partition the training task in multiple training subtasks with submodels, which can be performed independently and in parallel [8], to use asynchronous averaged stochastic gradient descent [9], and to reduce data transmission through a sampling-based approach [10]. Besides, in [11], the authors studied how the elimination of "unfavorable" samples improve generalization accuracy for convolutional neural networks. Finally, in [12], an unweighted influence data subsampling method is proposed.

Roughly speaking, a training process consists of searching for a local minimum of a loss function in an abstract space where the *states* are sets of weights. Each of the training sample batches provides an extremely small change in weights according to the training rules. The aim of such changes is to find the "best" set of weights that minimizes the loss function. If we consider a

R. Gonzalez-Diaz, M. A. Gutiérrez-Naranjo and E. Paluzo-Hidalgo have contributed equally to this work.

✉ Eduardo Paluzo-Hidalgo
epaluzo@us.es

Rocio Gonzalez-Diaz
rogodi@us.es

Miguel A. Gutiérrez-Naranjo
magutier@us.es

[1] Dept. of Applied Mathematics I, Universidad de Sevilla, 41012 Sevilla, Spain

[2] Dept. of Computer Science and Artificial Intelligence, Universidad de Sevilla, 41012 Sevilla, Spain

Published online: 05 May 2022

"geometrical" interpretation of the learning process, such changes can be seen as tiny steps in a multi-dimensional metric space of parameters that follow the direction settled by the gradient of the loss function. In some sense, one may think that two "close" points of the dataset with the same label provide "similar" information to the learning process since the gradient of the loss function on such points is similar.

Such a 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 "topological information," allowing the neural network to perform the learning process taking less time without losing accuracy. Proving such a general result for any neural network architecture 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. Due to such difficulties, we will begin in this paper by proving the usefulness of representative datasets in the perceptron case; that is, we formally prove, for the perceptron case, that the accuracy of a neural network evaluated on the representative dataset is similar to the accuracy of the neural network evaluated on the original dataset under some constraints on the representativeness of the dataset. Besides, experimental evidence indicates that neural networks trained on representative datasets perform similar to neural networks trained on the original datasets in the case of multi-layer neural networks.

Moreover, in order to "keep the shape" of the original dataset, the concept of representative datasets is associated with a notion of nearness independent of isometric transformations. As a first approach, the Gromov–Hausdorff distance is used to measure the *representativeness* of the dataset. Nonetheless, as the Gromov–Hausdorff distance complexity is an open problem[1], the bottleneck distance between persistence diagrams [14] is used instead as a lower bound to the Gromov–Hausdorff distance since its time complexity is cubic on the size of the dataset (see [15]).

The paper is organized as follows. In Sect. 2, basic definitions and results from neural networks and computational topology are given. The notion of representative datasets is introduced in Sect. 3. Persistence diagrams are used in Sect. 4 to measure the representativeness of a dataset. In Sect. 5, the perceptron architecture is studied to theoretically prove that the accuracy of a perceptron evaluated on original and representative datasets coincide under some constraint. In Sect. 6.1, experimental results are provided for the perceptron case showing the good performance of representative datasets, compared to random datasets. In Sect. 6.2, we illustrate experimentally the same fact for multi-layer neural networks. Finally, some conclusions and future work are provided in Sect. 7.

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], *Bidirectional Feature Pyramid Networks* [16] or new variants of the *Gated Recurrent Units* [17, 18]), so the current notion of neural network is far from the classic multi-layer perceptron or radial basis function networks [19].

As a general setting, a neural network is a mapping $\mathcal{N}_{w,\Phi} : \mathbb{R}^n \to [\![0,k]\!]$ (where $[\![0,k]\!] = \{0,1,\ldots,k\}$) that depends on a set of weights w and a set of parameters Φ which involve 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 point x lies in $X \subset \mathbb{R}^n$ and label c_x lies in $[\![0,k]\!]\}$, for a fixed integer $k \in \mathbb{N}$. Observe that it should be satisfied that a point cannot have different labels. The sets X and $[\![0,k]\!]$ are called, respectively, the set of points and the set of labels in $\mathcal{D}$.

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.

Finally, let us introduce the concept of *accuracy* as a measure to evaluate the performance of a neural network.

Definition 1 The accuracy of a neural network $\mathcal{N}_{w,\Phi}$ evaluated on a dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^n$ and $c_x \in [\![0,k]\!]\}$, is defined as:

$$\mathbb{A}(\mathcal{D}, \mathcal{N}_w) = \frac{1}{|X|} \sum_{x \in X} I_w(x),$$

where, for any $x \in X$,

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

[1] It seems to be intractable in practice [13].

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 [14].

Homology provides mathematical formalism to count *holes* where holes refer to connected components, tunnels, cavities, and so on, being the q-dimensional homology group the mathematical representation for the q-dimensional holes in a given space. Persistent homology is usually computed when the homology cannot be determined. An example of the latter appears when a surface is sampled by a point cloud.

Persistent homology is based on the concept of *filtration*, which is an increasing sequence of *simplicial complexes*. The building blocks of a simplicial complex are q-simplices, being a 0-simplex a point, a 1-simplex a line segment, a 2-simplex a triangle, a 3-simplex a tetrahedron, and so on. An example of filtration is the Vietoris–Rips filtration (see [20]) that is built by successively increasing the radius of the balls centered at the points of a given set in an n-dimensional space and joining those vertices (points) whose balls intersect forming new simplices.

We say that a q-dimensional hole *is born* when it appears at a specific time along the filtration and it *dies* when it merges with another q-dimensional hole at a specific time along the filtration. One of the common graphical representations of births and deaths of the q-dimensional holes over time is the so-called (q-dimensional) *persistence diagram* which consists of a set of points on the Cartesian plane. This way, a point of a persistence diagram represents the birth and the death of a hole. Since deaths happen 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 that survive for a long time. The so-called bottleneck distance can be used to compare two persistence diagrams.

Definition 2 The (q-dimensional) bottleneck distance between two (q-dimensional) persistence diagrams Dgm and $\widetilde{\mathrm{Dgm}}$ is:

$$d_B(\mathrm{Dgm}, \widetilde{\mathrm{Dgm}}) = \inf_{\phi} \sup_{\alpha} \|\alpha - \phi(\alpha)\|_{\infty}$$

where $\alpha \in \mathrm{Dgm}$ and ϕ is any possible bijection between $\mathrm{Dgm} \cup \Delta$ and $\widetilde{\mathrm{Dgm}} \cup \Delta$, being Δ the set of points in the diagonal axis.

An useful result used in this paper is the following one that connects the Gromov–Hausdorff distance between two metric spaces and the bottleneck distance between the persistence diagrams obtained from their corresponding Vietoris–Rips filtrations. For the sake of brevity, the (q-dimensional) persistence diagram obtained from the Vietoris–Rips filtration computed from a subset X of $\mathbb{R}^n$, with $q \leq n$, will be simply called the (q-dimensional) persistence diagram of X and denoted by $\mathrm{Dgm}_q(X)$.

Theorem 1 [21, *Theorem* 5.2] *For any two subsets X and Y of $\mathbb{R}^n$, and for any dimension $q \leq n$, the bottleneck distance between the persistence diagrams of X and Y is bounded by the Gromov–Hausdorff distance of X and Y:*

$$d_B(\mathrm{Dgm}_q(X), \mathrm{Dgm}_q(Y)) \leq 2 d_{GH}(X, Y).$$

Let us recall that $d_{GH}(X,Y) = \frac{1}{2}\inf_{f,g}\left\{ d_H(f(X), g(Y)) \right\}$, where $d_H(X,Y) = \max\left\{ \sup_{x\in X} \inf_{y\in Y} \|x-y\|, \sup_{y\in Y} \inf_{x\in X} \|x-y\| \right\}$, and $f : X \to Z$ (resp. $g : Y \to Z$) denotes an isometric transformation of X (resp. Y) into some metric space Z. Summing up, we can conclude that

$$\frac{1}{2} d_B(\mathrm{Dgm}_q(X), \mathrm{Dgm}_q(Y)) \leq d_{GH}(X,Y) \leq d_H(X,Y).$$

3 Representative datasets

In this section, we provide the definition of representative datasets which is independent of the neural network architecture considered. The intuition behind this definition is to keep the "shape" of the original dataset while reducing its number of points.

Firstly, let us introduce the notion of ε-representative point.

Definition 3 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 $\|x - \tilde{x}\| \leq \varepsilon$, 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 can also be correctly classified by the neural network (after adjusting the weights). Therefore, the definition of ε-representative dataset should be independent of such transformations. The concept of λ-balanced ε-representative datasets is also introduced and it will be used in Sect. 5 to ensure that similar accuracy results are obtained when a trained perceptron is evaluated on a representative dataset rather than on the original dataset.

Definition 4 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 $f(\tilde{x}) \approx_\varepsilon x$. The minimum of all those possible ε for which an isometric transformation exists is called to be *optimal*. Finally, the dataset $\tilde{\mathcal{D}}$ is said to be λ-balanced if for each $(\tilde{x}, c_{\tilde{x}}) \in \tilde{\mathcal{D}}$, the set $\{(x, c_x) : f(\tilde{x}) \approx_\varepsilon x\}$ contains λ points and for each $(x, c_x) \in \mathcal{D}$ there exists only one $(\tilde{x}, c_{\tilde{x}}) \in \tilde{\mathcal{D}}$ such that $f(\tilde{x}) \approx_\varepsilon x$.

Remark 1 Let us point out that λ-balanced datasets cannot be computed for most datasets when they are required to be subsets of the datasets. Even if it is too restrictive, in Sect. 5, we will provide several theoretical results using that assumption. Nevertheless, we will prove experimentally in Sect. 6 that, without that assumption, ε-representative datasets still perform well.

Proposition 1 *Let $\tilde{\mathcal{D}}$ be an ε-representative dataset (with set of points $\tilde{X} \subset \mathbb{R}^n$) of a dataset $\mathcal{D}$ (with set of points $X \subset \mathbb{R}^n$). Then,*

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

Proof By definition of ε-representative datasets, 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 the definition of the Gromov–Hausdorff distance, $d_{GH}(X, \tilde{X}) \leq d_H(X, f(\tilde{X})) \leq \varepsilon$. □

Notice that the definition of ε-representative datasets is not useful when ε is "big." The following result, which is a consequence of Proposition 1, provides the optimal value for ε.

Corollary 1 *The parameter ε is optimal if and only if $\varepsilon = d_{GH}(X, \tilde{X})$.*

Therefore, one way to discern if a dataset $\tilde{\mathcal{D}}$ is "representative enough" of $\mathcal{D}$ is to compute the Gromov–Hausdorff distance between X and $\tilde{X}$. If the Gromov–Hausdorff distance is "big," we could say that the dataset $\tilde{\mathcal{D}}$ is not representative of $\mathcal{D}$. However, the Gromov–Hausdorff distance is not useful in practice because of its high computational cost. An alternative approach to this problem is given in Sect. 4.

3.1 Proximity graph algorithm

In this section, for a given $\varepsilon > 0$, we propose a variant of the proximity graph algorithm [22] to compute an ε-representative dataset $\tilde{\mathcal{D}}$ of a dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^n$ and $c_x \in [\![0, k]\!]\}$.

Firstly, a proximity graph is built over X, establishing adjacency relations between the points of X, represented by edges.

Definition 5 Given $\varepsilon > 0$, 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 $(x, y) \in E$.

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

Secondly, from an ε-proximity graph of X, a dominating dataset (also known as a vertex cover) $\tilde{X} \subseteq X$ is computed satisfying that if $x \in X$ then $x \in \tilde{X}$ or there exists $y \in \tilde{X}$ adjacent to x. We then obtain an ε-representative dataset $\tilde{\mathcal{D}} = \{(\tilde{x}, c_{\tilde{x}}) : \tilde{x} \in \tilde{X}$ and $(\tilde{x}, c_{\tilde{x}}) \in \mathcal{D}\}$ also called *dominating dataset* of $\mathcal{D}$. Algorithm 1 shows the pseudo-code used in this paper to compute a dominating dataset of $\mathcal{D}$.

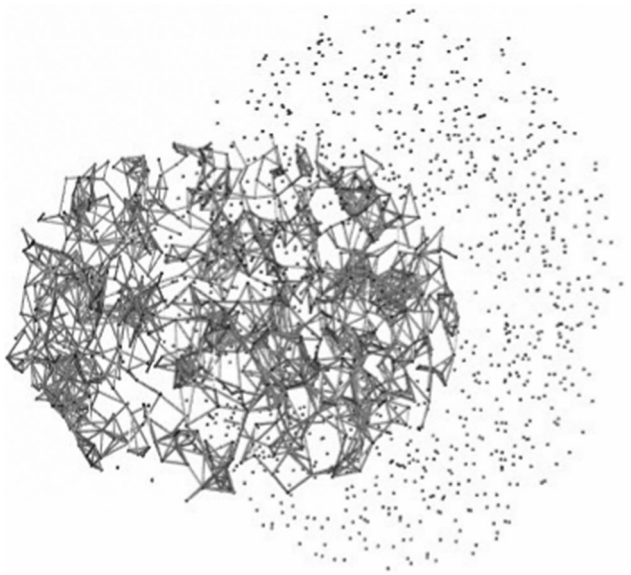


Fig. 1 A point cloud sampling two interlaced solid torus and the ε-proximity graph of one of them for a fixed ε.

Algorithm 1 Dominating Dataset Algorithm

Input: A dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^n \text{ and } c_x \in [\![0, k]\!]\}$ and a parameter $\varepsilon > 0$

Output: A dataset $\tilde{\mathcal{D}} \subseteq \mathcal{D}$

1: **for** $c = 0$ **to** $c = k$ **do**
2: $\quad X_c = \{x : (x, c) \in \mathcal{D}\}$
3: $\quad \tilde{X}_c = \text{DominatingSet}(G_\varepsilon(X_c))$
4: **end for**
5: $\tilde{X} = \cup_{c=0}^{k} \tilde{X}_c$
6: $\tilde{\mathcal{D}} = \{(x, c) : x \in \tilde{X} \text{ and } (x, c) \in \mathcal{D}\}$

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 Sect. 6, the algorithm proposed in [23] 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 complexity of the algorithm to obtain the dominating set. Let us observe that the algorithm proposed in this paper is just an example to show how we can compute representative datasets. Other more efficient algorithms to compute representative datasets are left for future work in Sect. 7.

Lemma 1 *The dominating dataset $\tilde{\mathcal{D}}$ obtained by running Algorithm* 1 *is an 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 arise:

1. If $(x, c_x) \in \tilde{\mathcal{D}}$, it is done.
2. If $(x, c_x) \notin \tilde{\mathcal{D}}$, since $\tilde{X}_{c_x}$ is a dominating dataset 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 Persistent homology to Infer the representativeness of a dataset

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

Firstly, from Theorem 1 in page 5, we will establish that the bottleneck distance between persistence diagrams is a lower bound of the representativeness of the dataset.

Lemma 2 *Let $\tilde{\mathcal{D}}$ be an ε-representative dataset (with set of points $\tilde{X} \subset \mathbb{R}^n$) of a dataset $\mathcal{D}$ (with set of points $X \subset \mathbb{R}^n$). Let* $\text{Dgm}_q(X)$ *and* $\text{Dgm}_q(\tilde{X})$ *be the q-dimensional persistence diagrams of X and $\tilde{X}$, respectively. Then, for $q \leq n$,*

$$\frac{1}{2} d_B\big(\text{Dgm}_q(X), \text{Dgm}_q(\tilde{X})\big) \leq \varepsilon.$$

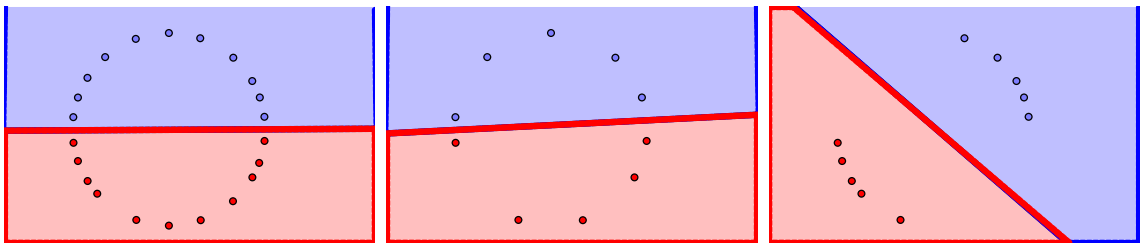
Proof Since $\tilde{\mathcal{D}}$ is an ε-representative dataset of $\mathcal{D}$ then $d_{GH}(X, \tilde{X}) \leq \varepsilon$ by Proposition 1. Now, by Theorem 1, $\frac{1}{2} d_B(\text{Dgm}_q(X), \text{Dgm}_q(Y)) \leq d_{GH}(X, Y) \leq \varepsilon$. □

As a direct consequence of Lemma 2 and the fact that the Hausdorff distance is an upper bound of the Gromov–Hausdorff distance, we have the following result.

Corollary 2 *Let $\tilde{\mathcal{D}}$ be an ε-representative dataset (with set of points $\tilde{X} \subset \mathbb{R}^n$) of a dataset $\mathcal{D}$ (with set of points $X \subset \mathbb{R}^n$) where the parameter ε is optimal. Let* $\text{Dgm}_q(X)$ *and* $\text{Dgm}_q(\tilde{X})$ *be the q-dimensional persistence diagrams of X and $\tilde{X}$, respectively. Then,*

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

In order to illustrate the usefulness of this last result, we will discuss a simple example. In Fig. 2a, we can see a subsample of a circumference (the original dataset) together with two classes corresponding, respectively, to the upper and lower part of the circumference. In Fig. 2c, we can see a subset of the original dataset and a decision boundary "very" different to the one given in Fig. 2a. Then, we could say that the dataset shown in Fig. 2c does not "represent" the same classification problem than the original dataset. However, the dataset shown in Fig. 2b could be considered a representative dataset of the original one since both decision boundaries are "similar." This can be determined by computing the Hausdorff distance between the original and the other datasets, and the bottleneck distance between the persistence diagrams of the corresponding datasets (see the values shown in Table 1). Using Corollary 2, we can infer that $0.08 \leq \varepsilon_1 \leq 0.18$ for



(a) A binary classification problem given by a sampled circumference. In this case, the classification problem tries to distinguish between the upper and the lower part of the circumference.

(b) **(ε_1-Representative dataset)** A subset of the sampled circumference given in Fig. 2a. Let us observe that the decision boundary obtained is similar to the one showed in Fig. 2a.

(c) **(ε_2-Representative dataset)** A subset of the sampled circumference given in Fig. 2a. Let us observe that the decision boundary obtained is quite different to the one showed in Fig. 2a.

Fig. 2 Illustration of a binary classification problem and the representative dataset concept

Table 1 The 0-dimensional bottleneck distance (d_{B0}), the 1-dimensional bottleneck distance (d_{B1}), and the Hausdorff distance (d_H) between the persistence diagrams of the dataset given in Fig. 2a and the datasets given in Fig. 2b and c, respectively

Datasets	$\frac{1}{2}d_{B0}$	$\frac{1}{2}d_{B1}$	d_H
Original (Fig. 2a) and ε_1-representative (Fig. 2b)	0.07	0.08	0.18
Original (Fig. 2a) and ε_2-representative (Fig. 2c)	0.13	0.08	0.3

the dataset given in Fig. 2b and $0.13 \leq \varepsilon_2 \leq 0.3$ for the dataset given in Fig. 2c. Therefore, the dataset given in Fig. 2b can be considered "more" representative of the dataset shown in Fig. 2a than the dataset given in Fig. 2c, as expected.

In Table 2, we show the output of Algorithm 1 applied to the dataset pictured in Fig. 2a and different values of ε, in order to obtain different dominating datasets. Let us observe that, depending on the value of the parameter ε, the size of the resulting dominating dataset is different. Finally, let us remark that the values of the parameter used in Algorithm 1 does not correspond, in general, to the optimal ε (see Definition 4).

Table 2 The 0-dimensional bottleneck distance (d_{B_0}) and the Hausdorff distance (d_H) between the persistence diagrams of the dataset ($\mathcal{D}$) given in Fig. 2a, composed of 22 points, and the dominating datasets ($\tilde{\mathcal{D}}$) obtained applying Algorithm 1 for different values of ε, and a random dataset ($\mathcal{R}$) of the same size than the corresponding dominating dataset

ε (Alg. 1)	$\lvert\tilde{\mathcal{D}}\rvert$	$\frac{1}{2}d_{B_0}(\mathcal{D},\tilde{\mathcal{D}})$	$\frac{1}{2}d_{B_0}(\mathcal{D},\mathcal{R})$	$d_H(\mathcal{D},\tilde{\mathcal{D}})$	$d_H(\mathcal{D},\mathcal{R})$
0.6	8	0.22	0.29	0.39	0.83
0.3	14	0.13	0.16	0.23	0.7
0.2	18	0.05	0.15	0.19	0.71

5 Theoretical results on the perceptron case

One of the simplest neural network architecture is the perceptron. Our goal in this section is to formally prove that the accuracy of a perceptron evaluated on the original dataset and on its representative dataset are equivalent when we impose certain conditions on the representativeness.

For the sake of simplicity, in this section, we will restrict our interest to a binary classification problem, although our approach is valid for any classification problem. Therefore, our input is a *binary* dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^n$ and $c_x \in \{0, 1\}\}$. Besides, we will assume in this section that the training process tries to minimize the following error function:

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

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, called the mean squared error (MSE). An example of such a training process is the *gradient descent* training algorithm.

First, let us introduce the definition of a perceptron.

Definition 6 A *perceptron* $\mathcal{N}_w : \mathbb{R}^n \to \{0,1\}$, with weights $w = (w_0, w_1, \ldots, w_n) \in \mathbb{R}^{n+1}$, 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 $y_w : \mathbb{R}^n \to (0,1)$ defined as

$$y_w(x) = \sigma(wx)$$

where, for $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$,

$$wx = w_0 + w_1 x_1 + \ldots + w_n x_n,$$

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

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

is the *sigmoid function*.

Remark 2 In the previous definition, let us point out that the condition $y_w(x) \geq \frac{1}{2}$ is the same as the condition $wx \geq 0$ that usually appears in the definition of perceptron.

A useful property of the sigmoid function is the easy expression of its derivative. Let σ^m denote the composition $\sigma^m = \sigma^{m-\text{times}} \cdots \sigma$.

Lemma 3 *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 Firstly, let us observe that $(\sigma^m)'(z) = m\sigma^m(z)(1-\sigma(z)) > 0$ since $0 < \sigma(z) < 1$ for all $z \in \mathbb{R}$. Secondly, 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, $(\sigma^m)'(\log(m)) = \left(\frac{m}{m+1}\right)^{m+1}$ concluding the proof. □

From now on, we will consider that the associated isometric transformation f by which the dataset is ε-representative is applied to the representative dataset. Therefore, by abuse of notation, $\tilde{x}$ will mean $f(\tilde{x})$. Analogously, $\tilde{X}$ will mean $f(\tilde{X})$ and $\tilde{\mathcal{D}}$ will mean $f(\tilde{\mathcal{D}})$.

In the following lemma, we prove that the difference between the outputs 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 4 *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 the 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 3, 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,

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

Applying now the Hölder 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 result is a direct consequence of Lemma 3 and Lemma 4.

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

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

The next result establishes under which conditions ε-representative points are classified under the same label as the points they represent.

Lemma 5 *Let $\tilde{\mathcal{D}}$ be an ε-representative dataset of the binary dataset $\mathcal{D}$. Let $\mathcal{N}_w$ be a perceptron with weights $w \in \mathbb{R}^{n+1}$. Let $(x, c) \in \mathcal{D}$ and $(\tilde{x}, c) \in \tilde{\mathcal{D}}$ with $\tilde{x} \approx_\varepsilon x$. If $\varepsilon \leq \frac{\|wx\|}{\|w\|}$ then*

$$\mathcal{N}_w(x) = \mathcal{N}_w(\tilde{x}).$$

Proof First, if $wx = 0$, then $\varepsilon = 0$, therefore $x = \tilde{x}$ and then $\mathcal{N}_w(x) = \mathcal{N}_w(\tilde{x})$. Now, let us suppose that $wx < 0$. Then, $y_w(x) < \frac{1}{2}$ and $\mathcal{N}_w(x) = 0$ by the definition of

perceptron. Since $\varepsilon < \frac{\|wx\|}{\|w\|}$ then x and $\tilde{x}$ belong to the same semispace in which the space is divided by the hyperplane $wx = 0$. Therefore, $w\tilde{x} < 0$, then $y_w(\tilde{x}) < \frac{1}{2}$ and finally $\mathcal{N}_w(\tilde{x}) = 0$. Similarly, if $wx > 0$, then $\mathcal{N}_w(x) = 1 = \mathcal{N}_w(\tilde{x})$, concluding the proof. □

By Lemma 5, we can state that if ε is "small enough" then the perceptron $\mathcal{N}_w$ evaluated on $\mathcal{D}$ and $\tilde{\mathcal{D}}$ will coincide.

In the following results, we will restrict ourselves to λ-balance datasets as a theoretical convenience.

The next result relies on the accuracy (see Definition 1) of a perceptron evaluated on the original dataset and its representative dataset.

Theorem 2 *Let $\tilde{\mathcal{D}}$ be a λ-balanced ε-representative dataset of the binary dataset $\mathcal{D}$. Let $\mathcal{N}_w$ be a perceptron with weights $w \in \mathbb{R}^{n+1}$. If $\varepsilon \leq \min\left\{\frac{\|wx\|}{\|w\|} : (x, c_x) \in \mathcal{D}\right\}$ then*

$$\mathbb{A}(\mathcal{D}, \mathcal{N}_w) = \mathbb{A}(\tilde{\mathcal{D}}, \mathcal{N}_w).$$

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

$$\begin{aligned}\mathbb{A}(\mathcal{D}, \mathcal{N}_w) - \mathbb{A}(\tilde{\mathcal{D}}, \mathcal{N}_w) &= \frac{1}{|X|}\sum_{x\in X} I_w(x) - \frac{1}{|\tilde{X}|}\sum_{\tilde{x}\in\tilde{X}} I_w(\tilde{x}) \\ &= \frac{1}{|X|}\sum_{x\in X}(I_w(x) - \lambda \cdot I_w(\tilde{x})) \\ &= \frac{1}{|X|}\sum_{\tilde{x}\in\tilde{X}}\sum_{x\approx_\varepsilon\tilde{x}}(I_w(x) - I_w(\tilde{x})).\end{aligned}$$

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 5 since $\varepsilon < \frac{\|wx\|}{\|w\|}$ for all $(x, c_x) \in \mathcal{D}$. □

Next, let us compare the two errors $\mathbb{E}(w, \mathcal{D})$ and $\mathbb{E}(w, \tilde{\mathcal{D}})$ obtained when considering the binary dataset $\mathcal{D}$ and its λ-balanced ε-representative dataset $\tilde{\mathcal{D}}$.

Theorem 3 *Let $\tilde{\mathcal{D}}$ be a λ-balanced ε-representative dataset of the binary dataset $\mathcal{D}$. Then*:

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

where ρ_m (being $m = 1, 2$) was defined in Lemma 4, and for each addend, $x \approx_\varepsilon \tilde{x}$.

Proof First, let us observe that:

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

Now, since $\tilde{\mathcal{D}}$ is λ-balanced ε-representative of $\mathcal{D}$ then $|X| = \lambda \cdot |\tilde{X}|$. Therefore,

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

where, for each addend, $\tilde{x} \approx_\varepsilon x$. Applying Lemma 4 for $m = 1, 2$ to the last expression, we get:

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

□

From this last result, we can infer the following: We can always fix the parameter ε "small enough" so that the difference between the error obtained when considering the dataset $\mathcal{D}$ and its ε-representative dataset is "close" to zero. Fig. 3 aims to provide intuition for this result.

Theorem 4 *Let $\delta > 0$. Let $\tilde{\mathcal{D}}$ be a λ-balanced ε-representative dataset of the binary dataset $\mathcal{D}$. Let $\mathcal{N}_w$ be a perceptron with weights $w \in \mathbb{R}^{n+1}$. If $\varepsilon \leq \frac{54}{43\|w\|_*}\delta$, then*

$$\|\mathbb{E}(w, \mathcal{D}) - \mathbb{E}(w, \tilde{\mathcal{D}})\| \leq \delta.$$

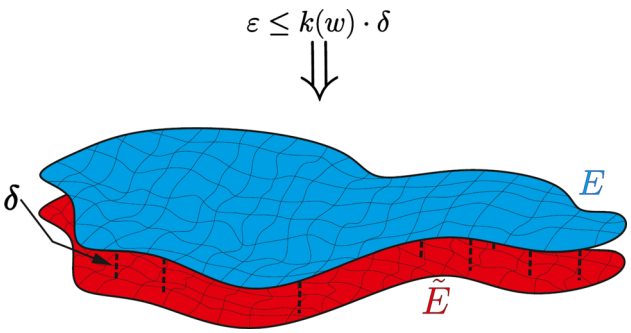


Fig. 3 Intuition for Theorem 4. The error function can be understood as an error surface. For a fixed set of weights w, the difference between the error computed on the original dataset, $E = \mathbb{E}(w, \mathcal{D})$, and on its λ-balanced ε-representative dataset, $\tilde{E} = \mathbb{E}(w, \tilde{\mathcal{D}})$, is bounded

Proof First, $\rho_1 \leq \frac{1}{4}$ and $\rho_2 \leq \frac{8}{27}$ by Corollary 3. Second, since $c_x \in \{0, 1\}$, we have:

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

Applying Hölder inequality to the last expression, we get:

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

Therefore, by Theorem 3, if $\varepsilon \leq \frac{54}{43\|w\|_*}\delta$, then $\|\mathbb{E}(w, X) - \mathbb{E}(w, \tilde{X})\| < \delta$ as stated. □

Summing up, we have proved that the accuracy and error of a perceptron evaluated on the binary dataset $\mathcal{D}$ or on its λ-balanced ε-representative dataset, are equivalent. This fact will be highlighted in Sect. 6.1 for the perceptron case and in Sect. 6.2 for neural networks with more complex architectures.

6 Experimental results

In this section, we experimentally prove that the accuracy of a neural network trained on the original dataset and on a dominating dataset is correlated with the parameter ε. Besides, we also show that the accuracy is worse if we train the neural network on a random dataset. Evaluation metrics will be computed to show the performance of the trained neural network using different datasets. Specifically, *MSE* is the mean squared error, *Recall* is the ratio of positive identifications correctly classified over all positive identifications. *Precision* is the ratio of positive identifications correctly classified over all those classified positive. *AUC* is the area under the ROC curve. The ROC curve plots true positive rates vs. false positive rates at different classification thresholds.

6.1 The perceptron case

In this section, two experiments are provided to support our theoretical results for the perceptron case and to illustrate the usefulness of our method.

In the first experiment (Sect. 6.1.1), several synthetic datasets are presented showing different distributions. Random weight initialization is considered, and the holdout procedure is applied (i.e., the datasets were split into training dataset and test set) to test the generalization capabilities.

In the second experiment (Sect. 6.1.2), the Iris dataset is considered. The perceptron is initiated with random weights and trained on three different datasets: the original dataset, a representative dataset (being the output of Algorithm 1) and a random dataset of the same size as the size of the representative dataset. Now, the trained perceptron is evaluated on the original dataset. This experiment supports that a perceptron trained on representative datasets get similar accuracy to a perceptron trained on 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 that representative datasets ensure good performance, while the random dataset provides no guarantees.

6.1.1 Synthetic datasets

In this experiment, different datasets were generated using a Scikit-learn Python package implementation.[2] Roughly speaking, it creates clusters of normally distributed points in an hypercube and adds some noise. Specifically, we considered three different situations: (1) distribution without overlapping; (2) distribution with overlapping; and (3) a dataset with a "thin" class and a high ε. In the last experiment, we wanted to show that the choice of ε is important, and that there are cases where representative datasets are not so useful. In all three cases, the perceptron was trained using the stochastic gradient descent algorithm and the mean squared error as the loss function.

The methodology followed in the experiments performed in this section is outlined in Fig. 4 and summarized in the following steps.

- Input: A dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^2$ and $c_x \in \{0, 1\}\}$ and a parameter $\varepsilon > 0$.
- Divide the dataset $\mathcal{D}$ in a training dataset $\mathcal{S}$ and a test set $\mathcal{T}$.
- Compute a dominating dataset $\tilde{\mathcal{S}}$ of $\mathcal{S}$ using Algorithm 1.
- Compute a random dataset $\mathcal{R}$ of $\mathcal{S}$.
- Train the perceptron $\mathcal{N}_w$ on $\mathcal{X}$, for $\mathcal{X} \in \{\mathcal{S}, \tilde{\mathcal{S}}, \mathcal{R}\}$.
- Evaluate the trained perceptron on $\mathcal{T}$.

The aim of the second step in the methodology is to compute an ε-representative dataset using Algorithm 1 (see Lemma 1 of Sect. 3.1). Nevertheless, using Algorithm 1 is not mandatory, and we could replace it by any other

[2] It can be found in https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html.

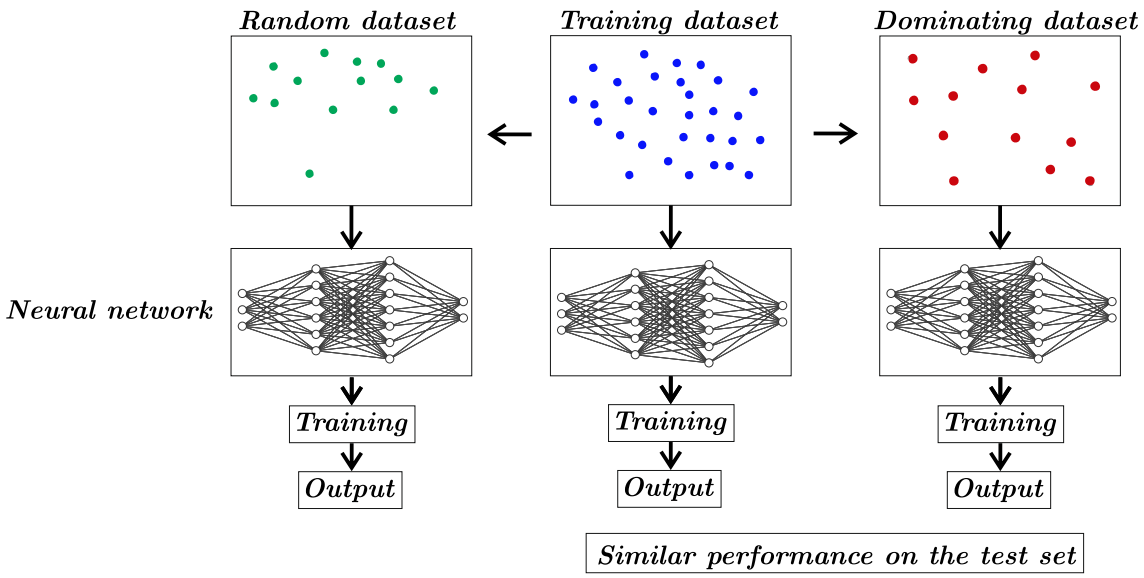


Fig. 4 Methodology followed in the experiments carried out in Sects. 6.1.1 and 6.2.1

process to compute an ε-representative dataset. Observe that the reduction is done in the fifth step since the idea is to train the neural network on the ε-representative dataset instead of on the training dataset.

In the first case (see Fig. 5a), 5000 points were taken with the two clusters well differentiated, i.e., without overlapping between classes. The 20% of the points were selected to belong to the test set, and the rest of the points constituted the training dataset. Then, an ε-representative dataset of the training dataset was computed using Algorithm 1 with $\varepsilon = 0.8$, obtaining a dominating dataset with just 17 points. Similarly, 17 random points were chosen from the training dataset (see Fig. 5c). Later, a perceptron was trained on each dataset for 20 epochs and evaluated on the test set. The mean accuracy results after 5 repetitions were: 0.96 for the dominating dataset, 0.82 for the random dataset, and 0.98 for the training dataset. Besides, the random dataset reached very low accuracy in general. In the second case (see Fig. 5d), a dataset composed of 5000 points with overlapping classes was generated. As in the first case, the dataset was split into a training dataset and a test set. Then, the ε-representative dataset of the training dataset was computed using Algorithm 1 with $\varepsilon = 0.5$, resulting in a dominating dataset of size 22. After training a perceptron for 20 epochs and repeating the experiments 5 times, the mean accuracy values were: 0.73 for the dominating dataset; 0.67 for the random dataset; and 0.86 for the training dataset. Finally, in the third case (see Fig. 5g), one of the classes was very "thin," in the sense that the points were very close to each other displaying a thin line. Therefore, if a "big" ε were chosen, that class would be represented by a pointed line as shown in Fig. 5h where $\varepsilon = 0.8$, reducing the dominating dataset to 15 points. With this example, we wanted to show a case where representative datasets were not so useful. The perceptron was trained for 20 epochs, and the mean accuracy of 5 repetitions was: 0.72 for the dominating dataset; 0.76 for the random dataset; and 0.99 for the training dataset. In terms of time, the training for 20 epochs on the training dataset took around 20 seconds, and the training on the dominating dataset took half a second. The computation of the dominating dataset took around 7 seconds. In Table 3, some evaluation metrics are provided on the test set.

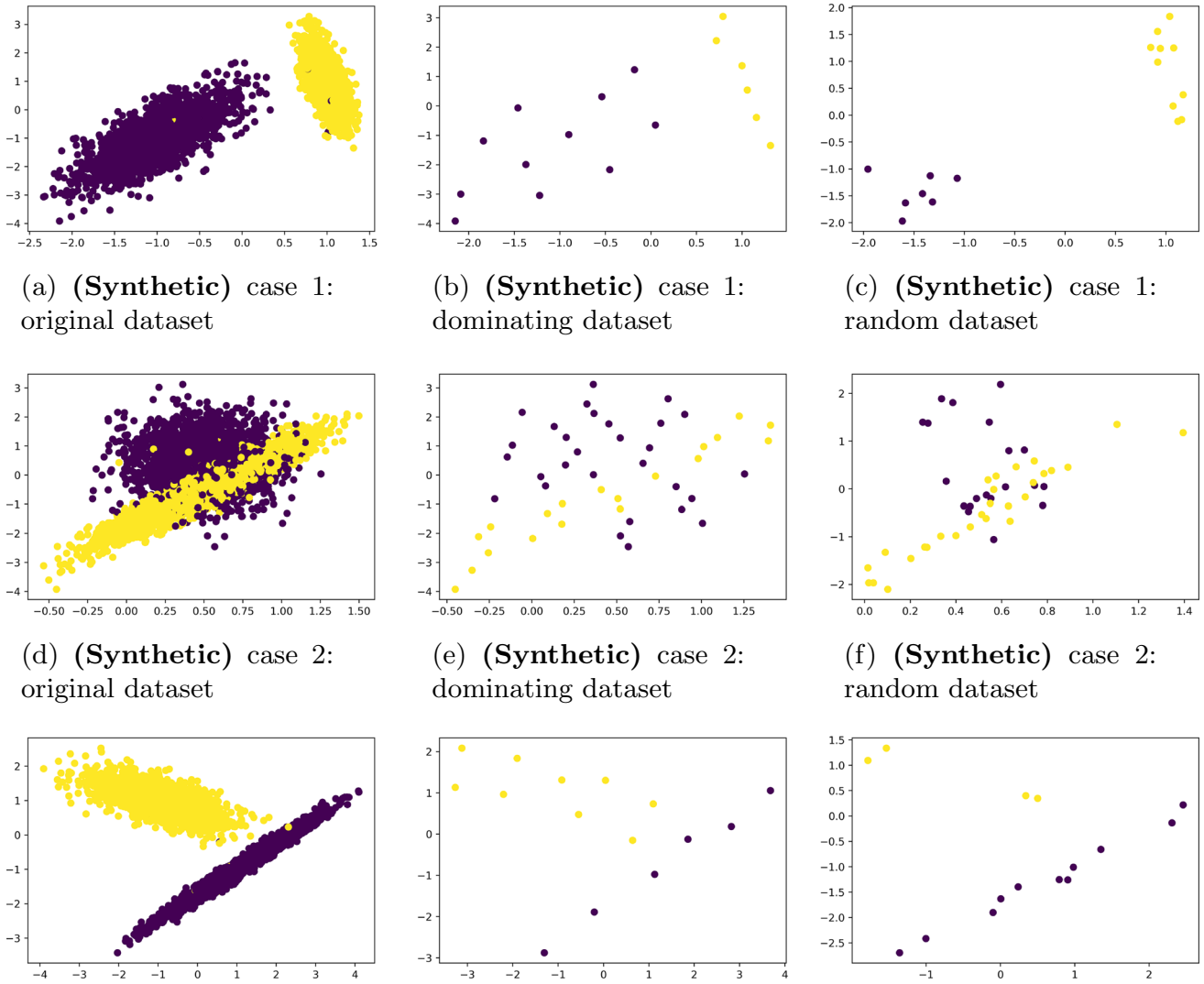
6.1.2 The iris dataset

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 that will be our original dataset.

The methodology followed in the experiments performed in this section is outlined in Fig. 6 and summarized in the following steps.

- Input: The original dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^4$ and $c_x \in \{0, 1\}\}$ and a parameter $\varepsilon > 0$.
- Compute a dominating dataset $\tilde{\mathcal{D}}$ of $\mathcal{D}$ using Algorithm 1.
- Compute a random dataset $\mathcal{R}$ of $\mathcal{D}$.
- Train a perceptron $\mathcal{N}_w$ on $\mathcal{X}$, for $\mathcal{X} \in \{\mathcal{D}, \tilde{\mathcal{D}}, \mathcal{R}\}$.
- Evaluate the trained perceptron on $\mathcal{D}$.

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



(a) **(Synthetic)** case 1: original dataset

(b) **(Synthetic)** case 1: dominating dataset

(c) **(Synthetic)** case 1: random dataset

(d) **(Synthetic)** case 2: original dataset

(e) **(Synthetic)** case 2: dominating dataset

(f) **(Synthetic)** case 2: random dataset

(g) **(Synthetic)** case 3: original dataset

(h) **(Synthetic)** case 3: dominating dataset

(i) **(Synthetic)** case 3: random dataset

Fig. 5 Different synthetic datasets generated using the Scikit-learn python package implementation. The first column corresponds to original datasets, the second column corresponds to dominating datasets of the training datasets, and the third column corresponds to random subsets of the training datasets of the same size as the corresponding dominating set

Table 3 Evaluation metrics on the test set for a perceptron trained on the training datasets, the dominating datasets and the random datasets computed from the synthetic datasets shown in Fig. 5

(Synthetic)	Dataset	Accuracy	Recall	Precision	AUC	MSE
Case 1	Training	0.98	0.99	0.98	0.99	0.01
	Dominating	0.96	0.97	0.97	0.99	0.09
	Random	0.82	0.96	0.82	0.84	0.15
Case 2	Training	0.86	0.85	0.93	0.92	0.1
	Dominating	0.73	0.89	0.72	0.79	0.18
	Random	0.67	0.93	0.66	0.78	0.2
Case 3	Training	0.99	0.98	0.99	0.99	0.01
	Dominating	0.72	0.87	0.64	0.95	0.18
	Random	0.76	0.71	0.61	0.71	0.22

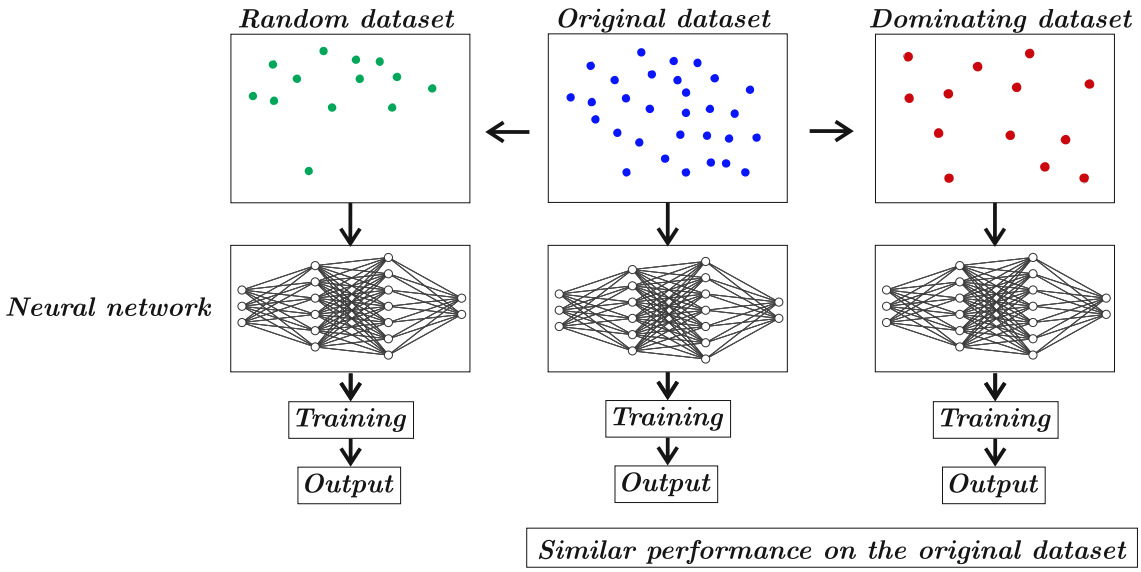
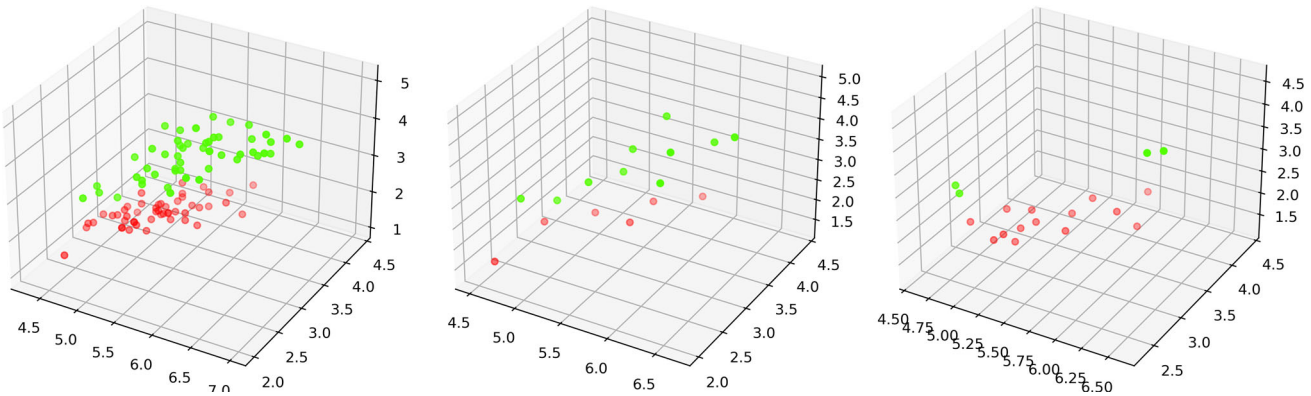


Fig. 6 Methodology followed in the experiments performed in Sects. 6.1.2 and 6.2.2.



(a) **(Iris)** original dataset

(b) **(Iris)** dominating dataset

(c) **(Iris)** random dataset

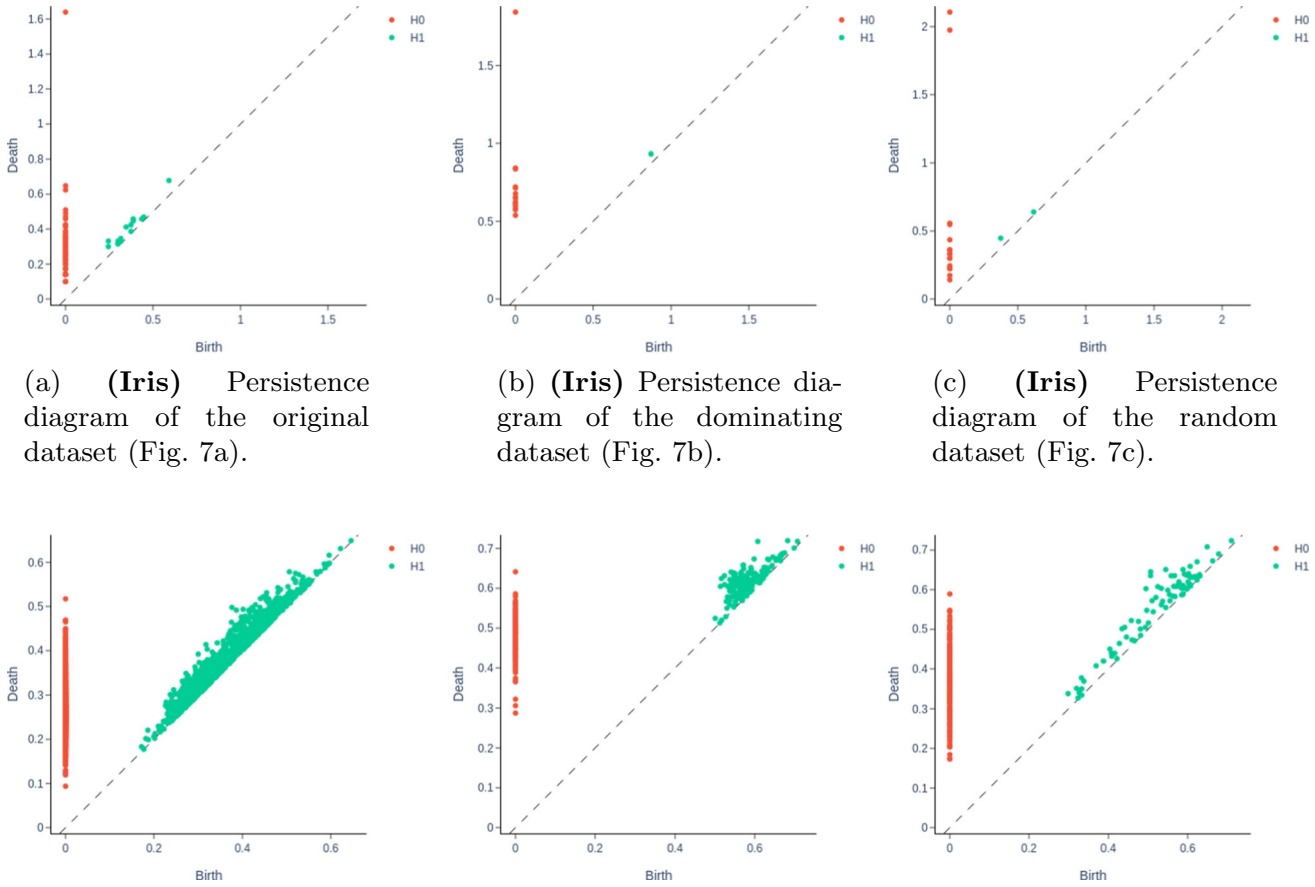
Fig. 7 Visualization of the Iris dataset: the original dataset is composed of 100 points, and the dominating dataset and the random dataset are composed of 16 points

Algorithm 1 was applied to the original dataset to obtain an ε-representative dataset of 16 points with $\varepsilon \leq 0.5$. A random dataset extracted from the original dataset with the same number of points than the dominating dataset was also computed. These datasets are represented in $\mathbb{R}^3$ in Fig. 7a, b and c, respectively. Besides, the associated persistence diagrams are shown in Fig. 8a, b and c. The Hausdorff and the 0-dimensional bottleneck distances between the original dataset, and the dominating and random datasets are given in Table 4.

We trained the perceptron with different initial weights and observed that the perceptron trained on the dominating and the original datasets converged to similar errors. In Table 5, the difference between the errors using a fixed set of weights for the dominating and the random dataset is provided. In Table 6, different metrics were evaluated on the original dataset when training on the original, the dominating and the random dataset, respectively. The table shows that the dominating dataset provides better metrics than the random dataset (Table 6). In Table 7, the computation time in seconds when using the different datasets is shown.

6.2 The multi-layer neural network case

In this section, we will experimentally check the usefulness of representative datasets for more complex neural network architectures than a perceptron. Two different experiments



(a) **(Iris)** Persistence diagram of the original dataset (Fig. 7a).

(b) **(Iris)** Persistence diagram of the dominating dataset (Fig. 7b).

(c) **(Iris)** Persistence diagram of the random dataset (Fig. 7c).

(d) **(Digits)** Persistence diagram of the original dataset (Section 6.2.2).

(e) **(Digits)** Persistence diagram of the dominating dataset (Section 6.2.2).

(f) **(Digits)** Persistence diagram of the random dataset (Section 6.2.2).

Fig. 8 Persistence diagram of the original, dominating and random datasets obtained from the Iris dataset and Digits dataset, respectively

Table 4 The Hausdorff distance (d_H) and the 0-dimensional bottleneck distance (d_B) of the dominating and random datasets with respect to the original datasets obtained from the Iris dataset and the Digits dataset, respectively

	Dataset	Size	$\frac{1}{2}d_B$	d_H
(Iris)	Original	100	–	–
	Dominating	16	0.11	0.58
	Random	16	0.5	1.12
(Digits)	Original	1797	–	–
	Dominating	173	0.09	0.29
	Random	173	0.09	0.31

were made, one using synthetic datasets and the other using the Digits dataset[4].

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

6.2.1 Synthetic datasets

This experiment consists of two different binary classification problems on synthetic datasets with 5000 points.

The methodology followed in the experiments performed in this section is outlined in Fig. 4 and summarized in the following steps.

- Input: A dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^3$ and $c_x \in \{0, 1\}\}$ and a parameter $\varepsilon > 0$.

Table 5 Comparison between the exact error differences computed over the random and the dominating datasets for the Iris classification problem

	Dataset	$\|E(w, X) - E(w, \tilde{X})\|$
(Iris)	Dominating	0.05
	Random	0.27

The values correspond to the mean of the exact error differences obtained using 100 different random weights

Table 6 Different metrics for the Iris and the Digits dataset experiments calculated as the mean values of 5 repetitions and evaluated on the original dataset

	Dataset	Accuracy	Recall	Precision	AUC	MSE
(Iris)	Original	1	1	1	0.99	0.003
	Dominating	0.9	1	0.9	0.9	0.11
	Random	0.6	0.2	0.2	0.36	0.39
(Digits)	Original	0.95	0.94	0.96	0.99	0.01
	Dominating	0.77	0.76	0.78	0.95	0.04
	Random	0.63	0.62	0.64	0.89	0.06

- Select a multi-layer neural network architecture $\mathcal{N}_{w,\Phi}$ to classify $\mathcal{D}$.
- Divide the dataset $\mathcal{D}$ in a training dataset $\mathcal{S}$ and a test set $\mathcal{T}$.
- Compute a dominating dataset $\tilde{\mathcal{S}}$ of $\mathcal{S}$ using Algorithm 1.
- Compute a random dataset $\mathcal{R}$ of $\mathcal{S}$.
- Train the multi-layer neural network $\mathcal{N}_{w,\Phi}$ on $\mathcal{X}$ for $\mathcal{X} \in \{\mathcal{S}, \tilde{S}, \mathcal{R}\}$.
- Evaluate the trained neural network on $\mathcal{T}$.

Each synthetic dataset (case A and case B) was split into a training dataset and a test set with proportions of 80% and 20%, respectively. Then, a dominating dataset of the training dataset, and a random subset of the training dataset with the same size as the dominating dataset, were computed and used for training a $3 \times 12 \times 6 \times 1$ neural network. It used ReLU activation function in the inner layers and sigmoid function in the output layer and was trained using stochastic gradient descent and mean squared error as the loss function for 20 epochs.

In the first case (see Fig. 9a), an unbalanced dataset with overlapping was considered, and an ε-dominating dataset was computed with $\varepsilon = 0.8$ composed of 67 points (see Fig. 9b). Then, a random dataset with the same size as the dominating dataset was considered. The multi-layer neural network was trained, and the mean accuracy values after 5 repetitions were: 0.85 for the dominating dataset; 0.74 for the random dataset; and 0.86 for the training dataset. In the second case, a balanced dataset with overlapping was considered (see Fig. 9d), and the same process as in the first case was carried out but with $\varepsilon = 0.3$, obtaining a dominating dataset of size 319. The mean accuracy values after 5 repetitions were: 0.92 for the dominating dataset; 0.91 for the random dataset; and 0.93 for the training dataset. In Table 8, different evaluation metrics on the test set for the two cases are shown.

Table 7 Time (in seconds) required to compute the dominating datasets using Algorithm 1 and time (in seconds) required for the training process on the Iris and Digits datasets

	Dataset	Alg. 1 time	Training time
(Iris)	Original	–	3.06 sec
	Dominating	0.04 sec	0.8 sec
(Digits)	Original	–	65.6 sec
	Dominating	0.57 sec	11.78 sec

The training method consists of the gradient descent algorithm for the Iris dataset experiment. In the case of the Digits dataset, a multi-layer neural network was trained for 1000 epochs using the Adam training algorithm

6.2.2 The digits dataset

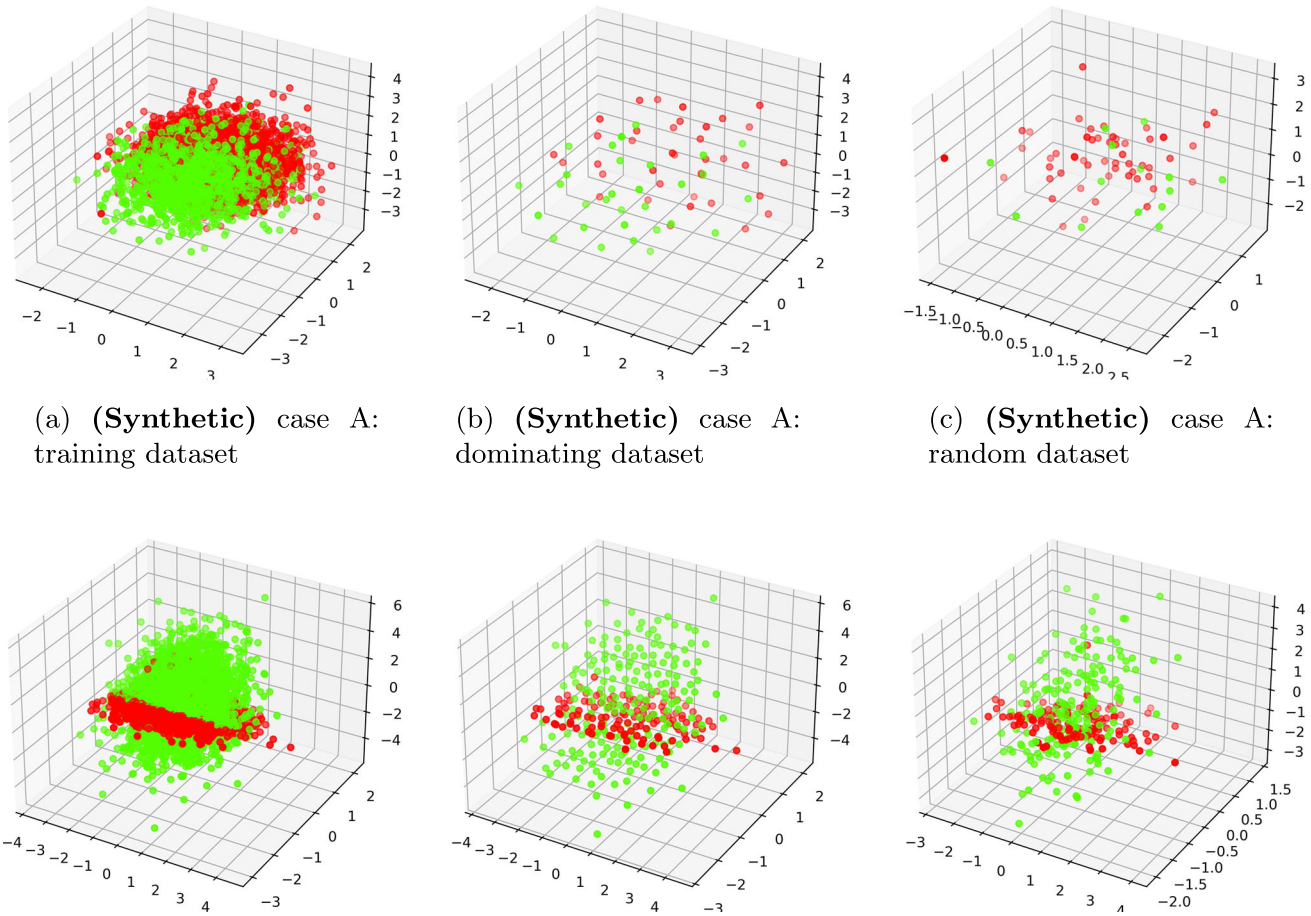
The Digits dataset[5] used in this experiment consists of images classified in 10 different classes corresponding to digits from 0 to 9. An example of an image of each class is shown in Fig. 10. The Digits dataset is composed by 1797 64-dimensional instances.

The methodology followed in the experiments performed in this section is outlined in Fig. 6 and summarized in the following steps.

- Input: A dataset $\mathcal{D} = \{(x, c_x) : x \in X \subset \mathbb{R}^{64}$ and $c_x \in \{0, 1, ..., 9\}\}$ and a parameter $\varepsilon > 0$.
- Select a multi-layer neural network architecture $\mathcal{N}_{w,\Phi}$ to classify $\mathcal{D}$.
- Compute a dominating dataset $\tilde{\mathcal{D}}$ of $\mathcal{D}$ using Algorithm 1.
- Compute a random dataset $\mathcal{R}$ of $\mathcal{D}$.
- Train the multi-layer neural network $\mathcal{N}_{w,\Phi}$ on $\mathcal{X}$ for $\mathcal{X} \in \{\mathcal{D}, \tilde{D}, \mathcal{R}\}$.
- Evaluate the trained neural network on $\mathcal{D}$.

In this experiment, Algorithm 1 was applied with $\varepsilon = 0.2$ to obtain a dominating dataset of size 173. The corresponding persistence diagrams can be seen in Fig. 8d, e and f. The Hausdorff and the bottleneck distances are shown in Table 4. In this case, we used a multi-layer neural network with $64 \times 400 \times 300 \times 800 \times 300 \times 10$ neurons with sigmoid activation function in the hidden layers and softmax activation function in the output layer. The neural network was trained using Adam algorithm and categorical cross-entropy as the loss function for 1000 epochs. It was launched 5 times for the dominating dataset and the random

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



(a) **(Synthetic)** case A: training dataset

(b) **(Synthetic)** case A: dominating dataset

(c) **(Synthetic)** case A: random dataset

(d) **(Synthetic)** case B: training dataset

(e) **(Synthetic)** case B: dominating dataset

(f) **(Synthetic)** case B: random dataset

Fig. 9 The first column shows the training datasets obtained as subsets of two different synthetic datasets generated using the Scikit-learn python package implementation. The second column shows dominating datasets computed from the training datasets, and the third column shows random subsets of the training datasets with the same size as the corresponding dominating dataset

Table 8 Evaluation metrics on the test set for the training of a multi-layer neural network on the training dataset, on the dominating dataset and on the random dataset obtained from the synthetic dataset experiment

(Synthetic)	Dataset	Accuracy	Recall	Precision	AUC	MSE
Case A	Training	0.86	0.61	0.85	0.88	0.11
	Dominating	0.85	0.64	0.8	0.84	0.14
	Random	0.74	0.05	0.38	0.73	0.18
Case B	Training	0.93	0.95	0.93	0.98	0.05
	Dominating	0.92	0.96	0.92	0.98	0.07
	Random	0.91	0.94	0.92	0.98	0.09

dataset. The mean accuracy values of different metrics for the 5 repetitions when training the neural network on the three different datasets and evaluated on the original dataset are shown in Table 6. Finally, in Table 9, different values for ε were used to compute the size of the dominating dataset and the accuracy of the neural network trained with the dominating dataset both on the dominating dataset and on the original dataset.

7 Conclusions and future work

The success of practical applications and the availability of new hardware (e.g., GPUs [24] and TPUs [25]) have led to focus neural network research on the development of new architectures rather than on theoretical issues. Nevertheless, a deeper understanding of the data structure is also necessary for field development, such as new discoveries on adversarial examples [26] have shown, or the one given

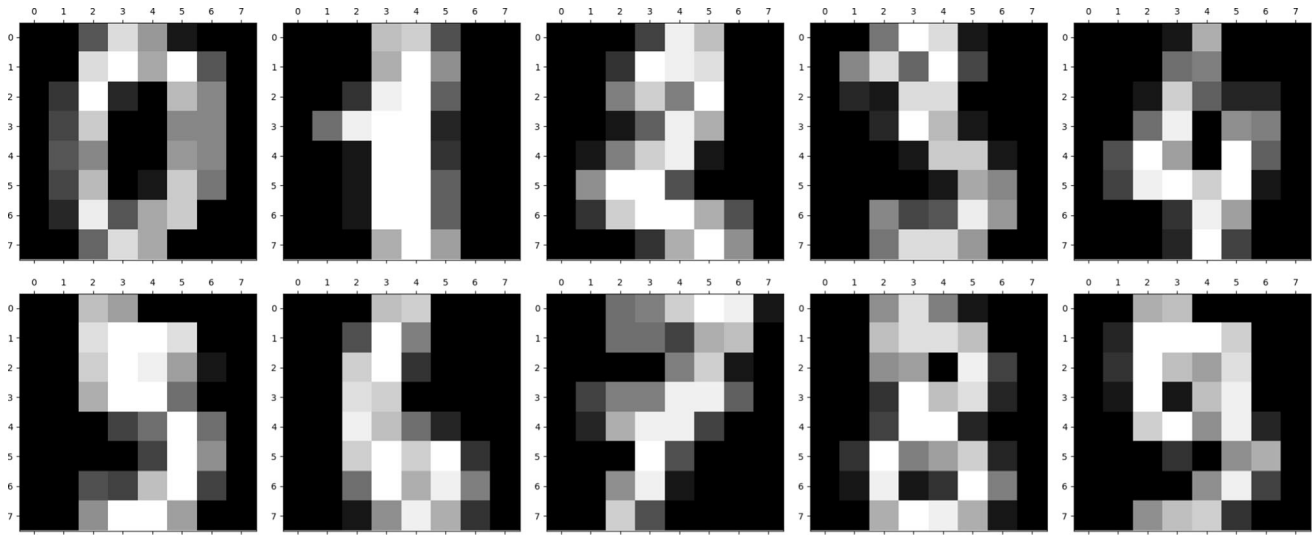


Fig. 10 **(Digits)** Example of an image of each class of the digits dataset. They are 32×32 arrays in gray scale

Table 9 Mean accuracy values after 5 repetitions of the neural network trained on the dominating dataset and evaluated on both the dominating dataset and the original dataset for the Digits classification problem

ε	Size	Accuracy on the original dataset	Accuracy on the dominating dataset
0.1	1363	0.95	0.95
0.15	531	0.91	0.98
0.18	283	0.82	0.99

in [27], where the redundancy of several datasets is empirically shown.

In this paper, we propose the use of *representative datasets* as a new approach to reduce learning time in neural networks based on the topological structure of the input dataset. Specifically, we have defined representative datasets using a notion of nearness that has the Gromov–Hausdorff distance as the lower bound. Nevertheless, the bottleneck distance of persistence diagrams (which is a lower bound of the Gromov–Hausdorff distance) is used to measure the representativeness of the dataset since it is computationally less expensive. Besides, we have theoretically proved that the accuracy of a perceptron evaluated on the original dataset coincides with the accuracy of the neural network evaluated on its representative dataset when the neural network architecture is a perceptron, the loss function is the mean square error and certain conditions on the representativeness of the dataset are imposed. Furthermore, the agreement between the provided theoretical results and the experiments supports that representative datasets can be a good approach to reach an efficient "summarization" of a dataset to train a neural network.

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 can be extended to other neural network architectures and training algorithms using milder constraints. Let us observe that in the case that the dataset is already small, and then, it will not make sense to compute a representative dataset. The sparsity of the dataset may be a feature to be considered as a future work. Finally, we plan to investigate more efficient ways of computing dominating datasets in particular and representative datasets in general, since the algorithm proposed is just an example and, in general, will not compute the smallest possible representative dataset nor is it the fastest possible.

Acknowledgements This work was partly supported by the Agencia Estatal de Investigación/10.13039/501100011033 under grant PID2019-107339GB-100 and the Agencia Andaluza del Conocimiento under grant P20-01145.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Declarations

Conflict of interest The authors declare that there is no conflict of interest.

Code availability The implementation of the methodology followed and the experimentation carried out can be consulted online in https://github.com/Cimagroup/Experiments-Representative-datasets.

References

1. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, MA. http://www.deeplearningbook.org
2. Mohammadi B, Fathy M, Sabokrou M (2021) Image/video deep anomaly detection: a survey. arXiv:2103.01739
3. Biswas R, Blanco-Medina P (2021) State of the art: face recognition. arXiv:2108.11821
4. Brown TB, Mann B, Ryder N, Subbiah M et al (2020) Language models are few-shot learners. In: Advances in neural information processing systems, vol. 33, pp. 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
5. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y (eds.) 3rd International Conference on Learning Representations, ICLR 2015. arXiv:1409.1556
6. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), pp 1–9. https://doi.org/10.1109/CVPR.2015.7298594
7. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778. https://doi.org/10.1109/CVPR.2016.90
8. Miranda CS, Zuben FJV (2015) Reducing the training time of neural networks by partitioning. arXiv:1511.02954
9. You Z, Xu B (2014) Improving training time of deep neural networkwith asynchronous averaged stochastic gradient descent. In: The 9th international symposium on chinese spoken language processing, pp 446–449. https://doi.org/10.1109/ISCSLP.2014.6936596
10. Xiao T, Zhu J, Liu T, Zhang C (2017) 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, pp 4193–4199. https://doi.org/10.24963/ijcai.2017/586
11. Wang T, Huan J, Li B (2018) Data dropout: optimizing training data for convolutional neural networks. In: 2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI), pp 39–46. https://doi.org/10.1109/ICTAI.2018.00017
12. Wang Z, Zhu H, Dong Z, He X, Huang S-L (2020) Less is better: unweighted data subsampling via influence function. Proc AAAI Conf Artif Intell 34(04):6340–6347. https://doi.org/10.1609/aaai.v34i04.6103
13. Schmiedl F (2017) Computational aspects of the gromov-hausdorff distance and its application in non-rigid shape matching. Discrete Comput Geom 57(4):854–880. https://doi.org/10.1007/s00454-017-9889-4
14. Edelsbrunner H, Harer JL (2010) Computational topology, an introduction. American Mathematical Society, Providence
15. Chazal F, Cohen-Steiner D, Guibas LJ, Mémoli F, Oudot SY (2009) Gromov-Hausdorff stable signatures for shapes using persistence. Computer Graphics Forum (proc. SGP 2009), 1393–1403
16. Tan M, Pang R, Le QV (2020) Efficientdet: scalable and efficient object detection. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 10778–10787.https://doi.org/10.1109/CVPR42600.2020.01079
17. Dey R, Salem FM (2017) Gate-variants of gated recurrent unit (gru) neural networks. In: 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), pp 1597–1600. https://doi.org/10.1109/MWSCAS.2017.8053243
18. Heck JC, Salem FM (2017) Simplified minimal gated unit variations for recurrent neural networks. In: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), pp 1593–1596. https://doi.org/10.1109/MWSCAS.2017.8053242
19. Haykin SS (2009) Neural networks and learning machines, 3rd edn. Pearson Education, Upper Saddle River
20. Hausmann J-C (2016) In: Quinn, F. (ed.) On the Vietoris-Rips complexes and a Cohomology Theory for metric spaces, pp 175–188. Princeton University Press, NJ. https://doi.org/10.1515/9781400882588-013
21. Chazal F, de Silva V, Oudot S (2014) Persistence stability for geometric complexes. Geometriae Dedicata 173(1):193–214. https://doi.org/10.1007/s10711-013-9937-z
22. Gonzalez-Diaz R, Paluzo-Hidalgo E, Gutierrez-Naranjo MA (2018) Representative datasets for neural networks. Electron Notes Discrete Math 68:89–94. https://doi.org/10.1016/j.endm.2018.06.016 (**Discrete Mathematics Days 2018**)
23. Matula DW (1987) Determining edge connectivity in 0(nm). In: 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), pp. 249–251. https://doi.org/10.1109/SFCS.1987.19
24. Gu J, Liu H, Zhou Y, Wang X (2017) DeepProf: performance analysis for deep learning applications via mining GPU execution patterns. arXiv:1707.03750
25. Jouppi NP, Young C, Patil N, Patterson D, et al (2017) In-datacenter performance analysis of a tensor processing unit. In: Proceedings of the 44th annual international symposium on computer architecture. ISCA '17, pp 1–12.https://doi.org/10.1145/3079856.3080246
26. Yuan X, He P, Zhu Q, Li X (2019) Adversarial examples: attacks and defenses for deep learning. IEEE Trans Neural Netw Learn Syst 30(9):2805–2824. https://doi.org/10.1109/TNNLS.2018.2886017
27. Toneva M, Sordoni A, Combes RTd, Trischler A, Bengio Y, Gordon GJ (2019) An empirical study of example forgetting during deep neural network learning. In: ICLR. https://openreview.net/forum?id=BJlxm30cKm