Journal of *Imaging*

MDPI

Article

Optimizing the Simplicial-Map Neural Network Architecture

Eduardo Paluzo-Hidalgo [1,*], Rocio Gonzalez-Diaz [1], Miguel A. Gutiérrez-Naranjo [2] and Jónathan Heras [3]

[1] Department of Applied Mathematics I, University of Sevilla, 41012 Sevilla, Spain; rogodi@us.es
[2] Department of Computer Sciences and Artificial Intelligence, University of Sevilla, 41012 Sevilla, Spain; magutier@us.es
[3] Department of Mathematics and Computer Science, University of La Rioja, 26004 Logroño, Spain; jonathan.heras@unirioja.es
* Correspondence: epaluzo@us.es

Abstract: Simplicial-map neural networks are a recent neural network architecture induced by simplicial maps defined between simplicial complexes. It has been proved that simplicial-map neural networks are universal approximators and that they can be refined to be robust to adversarial attacks. In this paper, the refinement toward robustness is optimized by reducing the number of simplices (i.e., nodes) needed. We have shown experimentally that such a refined neural network is equivalent to the original network as a classification tool but requires much less storage.

Keywords: simplicial-map neural networks; artificial neural networks; computational topology

Citation: Paluzo-Hidalgo, E.; Gonzalez-Diaz, R.; Gutiérrez-Naranjo, M.A.; Heras, J. Optimizing the Simplicial-Map Neural Network Architecture. *J. Imaging* **2021**, *7*, 173. https://doi.org/10.3390/jimaging7090173

Academic Editors: Matteo Rucco, Maurizio Mongelli, Anastasia Mavridou and Masoud Daneshtalab

Received: 29 June 2021
Accepted: 27 August 2021
Published: 1 September 2021

1. Introduction

In spite of the undoubted advantages of deep learning techniques for classification tasks [1], many important problems remain still unsolved in this context. In particular, if we focus on the efficiency of such models, one of their main drawbacks is the huge amount of resources needed for training competitive networks (for instance, vision models based on the transformer architecture need billions of images to be trained [2]). In many cases, only big companies can support the expensive cost of training competitive architectures [3,4]. From a practical point of view, one of the open research lines in deep learning is the exploration of ways to reduce training resources without reducing the accuracy of trained models.

One way to reduce time (to train the model) and space (to store the training data set) is to take a small subset of the training data set that summarizes its useful information. Several authors have explored this idea. For example, in [5], a data set representative of the training data set was considered. In [6], techniques of active learning were applied to classify images using convolutional neural networks. In [7], the authors reduced the volume of the training data set using stochastic methods. Other authors, in [8], replaced the training data set with a small number of synthetic samples containing all the original information.

Another approach tries to reduce the number of training parameters by pruning the model. This is a general technique in machine learning and it has a long tradition in neural networks [9]. The importance of pruning neural networks has emerged in recent years due to the big amount of resources required in deep learning [10–12]. Since local search techniques based on back propagation play a central role in weight optimization, the different pruning techniques can be classified using such a training process as the main reference. For example, there are studies where pruning occurs at the end of the training process [13], after the training process [14] or in the early stages of the training process [15].

Topological data analysis (TDA) provides a completely different approach to reducing the number of resources in the neural network classification process. In [16], the authors provided a constructive approach to the problem of approximating a continuous function on a compact set in a triangulated space. Once a triangulation of the space is given, a two-hidden-layer feedforward network with a concrete set of weights called a simplicial-map

neural network is computed. The construction is based on several strong theorems from algebraic topology and allows one to avoid the heavy process of optimizing the weights of neural networks since they can compute the weights directly from the triangulation of the space. Later, in [17], the authors showed that simplicial-map neural networks can be defined to be robust to adversarial attacks of a given size.

Simplicial-map neural networks are vaguely related to margin-based classifiers such as support vector machines (SVMs) and to nonparametric methods such as k-nearest neighbors (k-NN). These algorithms are widely used and, in both cases, there exist efforts to study their robustness to adversarial examples such as [18] in the case of k-NN or [19] for SVMs. Simplicial-map neural networks are not trained but defined on a triangulation of the data set and the decision boundaries are based on that triangulation. One of the greatest advantages of this approach is the possibility of formal proof of different properties such as universal approximation ability and, as previously mentioned, robustness against adversarial examples. However, both properties are based on barycentric subdivisions of the triangulation with a large increase in required storage as the number of simplices grows, this being a bottleneck for its applicability.

In this paper, we propose an algorithm to reduce the number of parameters of simplicial-map neural networks without reducing their accuracy. The key to the proposed method is that barycentric subdivisions, in particular, and triangulations of training data sets, in general, introduce many simplices that are not needed or redundant. The paper is organized as follows. In Section 2, we recall some basic concepts. In Section 3, we provide the description of our methodology. The description is illustrated with some examples in Section 4. We finish the paper with some conclusions and hints for future work.

2. Background

In [16,17], a new approach to construct neural networks based on simplicial maps was introduced. Roughly speaking, a combinatorial structure (a simplicial complex) K is built on top of a labeled data set using Delaunay triangulations to, lately, construct a neural network based on a simplicial map defined between K and a simplicial complex with just one maximal simplex. This section is devoted to recall some of the basic concepts used in such construction.

The research field of neural networks is exponentially growing and recently, many different architectures, activation functions, and regularization methods have been introduced; thus, it is difficult to find a general definition that covers all the cases. In this paper, we adapt a definition from [20] that fits into our purposes. From now on, n, m, d, k denote positive integers and $[\![1, n]\!]$ denote the set of integers $\{1, \ldots, n\}$.

Definition 1 (adapted from [20]). *A multilayer feedforward network defined between spaces $X \subseteq \mathbb{R}^d$ and $Y \subseteq \mathbb{R}^k$ is a function $\mathcal{N} : X \to Y$ composed of $m+1$ functions:*

$$\mathcal{N} = f_{m+1} \circ f_m \circ \cdots \circ f_1$$

where the integer $m > 0$ is the number of hidden layers and, for $i \in [\![1, m+1]\!]$, the function $f_i : X_{i-1} \to X_i$ is defined as

$$f_i(y) := \phi_i(W^{(i)}; y; b_i)$$

where $X_0 = X$, $X_{m+1} = Y$, and $X_i \subseteq \mathbb{R}^{d_i}$ for $i \in [\![1, m]\!]$; $d_0 = d$, $d_{m+1} = k$, and $d_i > 0$ being an integer for $i \in [\![1, m]\!]$ (called the width of the ith hidden layer); $W^{(i)} \in \mathcal{M}_{d_{i-1} \times d_i}$ being a real-valued $d_{i-1} \times d_i$ matrix (called the matrix of weights of $\mathcal{N}$); b_i being a point in $\mathbb{R}^{d_i}$ (called the bias term); and ϕ_i being a function (called the activation function). We will call the width of the neural network to the maximum width of hidden layers.

Throughout this paper, neural networks will be considered as classification models.

Definition 2. *A labeled data set D is a finite set of pairs*

$$D = \{(p_j, \ell_j) : j \in [\![1, n]\!],\ \ p_j \in \mathbb{R}^d,\ \ell_j \in \mathbb{E}^k\}$$

where, for $j, h \in [\![1, n]\!]$, $p_j \neq p_h$ if $j \neq h$, and ℓ_j represents a one-hot vector. We say that ℓ_j is the label of p_j or, equivalently, that p_j belongs to the class ℓ_j. We will denote by D_P the ordered set of points $\langle p_j \rangle_j$.

Given a data set and a set of neural networks that only differ in their weights, the supervised classification problem consists in finding an available neural network in the set that provides the best classification for the data set. Since neural networks in the set only differ in their weights, finding the best neural network is equivalent to find the best possible weights. Again, several definitions of the concept of supervised classification problem can be provided, mainly depending on the method used to look for the possible weights and the concept of improvement chosen to define the best option.

In this paper, the concept of supervised classification problem for neural networks is defined as follows.

Definition 3. *Given a labeled data set $D \subset \mathbb{R}^d \times \mathbb{E}^k$, an integer $m > 0$, and a set of activation functions ϕ_i for $i \in [\![1, m]\!]$, a supervised classification problem consists of looking for the weights $W^{(i)}$ and bias terms b_i for $i \in [\![1, m]\!]$, such that the associated neural network $\mathcal{N} : X \to Y$, with $X \subseteq \mathbb{R}^d$, $Y \subseteq \mathbb{R}^k$ and $D \subseteq X \times Y$, satisfies:*

- *$\mathcal{N}(p) = \ell$ for all $(p, \ell) \in D$.*
- *$\mathcal{N}$ maps $x \in X$ to a vector of scores $\mathcal{N}(x) = (y_1, \ldots, y_k) \in Y$ such that $y_i \in [0, 1]$ for $i \in [\![1, n]\!]$ and $\sum_{i \in [\![1,n]\!]} y_i = 1$.*

If such a neural network $\mathcal{N}$ exists, we will say that $\mathcal{N}$ characterizes D, or, equivalently, that $\mathcal{N}$ correctly classifies D.

The process to search for optimal weights is usually called the training of the neural network. The training most commonly used is based on backpropagation [21]. Nevertheless, in this paper, the optimal weights are not searched through an optimization process. Instead, a combinatorial structure is built on top of the training samples and a function called simplicial map is defined on it; then, a special kind of neural network named simplicial-map neural network is constructed. In order to recall the definition of simplicial-map neural network, we start by recalling the definitions of convex hull and convex polytope.

Definition 4. *The convex hull of a set $S \subset \mathbb{R}^d$, denoted by $\mathrm{conv}(S)$, is the smallest convex set containing S. If S is finite, then $\mathrm{conv}(S)$ is called a convex polytope and denoted by $\mathcal{P}$. The set of vertices of a convex polytope $\mathcal{P}$ is the minimum set $V_\mathcal{P}$ of points in $\mathcal{P}$ such that $\mathcal{P} = \mathrm{conv}(V_\mathcal{P})$.*

Our construction of simplicial-map neural networks is based on the simplicial complex obtained after a triangulation of the given convex polytope. Let us now recall the concept of simplicial complex.

Definition 5. *Let us consider a finite set V whose elements will be called vertices. A simplicial complex K consists of a finite collection of nonempty subsets (called simplices) of V such that:*

1. *Any subset of V with exactly one point of V is a simplex of K called 0-simplex or vertex.*
2. *Any nonempty subset of a simplex σ is a simplex, called a face of σ.*

A simplex σ with exactly $k + 1$ points is called a k-simplex. We also say that the dimension of σ is k and write $\dim \sigma = k$. A maximal simplex of K is a simplex that is not face of any other simplex in K. The dimension of K is denoted by $\dim K$ and it is the maximum dimension of its maximal simplices. The set of vertices of a simplicial complex K will be denoted by $K^{(0)}$. A simplicial complex K is pure if all its maximal simplices have the same dimension.

An example of simplicial complex is the Delaunay complex defined from the Voronoi diagram of a given finite set of points.

Definition 6. *Let $S = \{p_1, \ldots, p_n\}$ be a finite set of points in $\mathbb{R}^d$ in general position. The Voronoi cell $\mathcal{V}(p_i, S)$ is defined as:*

$$\mathcal{V}(p_i, S) := \{x \in \mathbb{R}^d : ||x - p_i|| \leq ||x - p_j||, \forall p_j \in S\}.$$

The Voronoi diagram of S, denoted as $\mathcal{V}(S)$, is the set of Voronoi cells:

$$\mathcal{V}(S) := \{\mathcal{V}(p_1, S), \ldots, \mathcal{V}(p_n, S)\}.$$

The Delaunay complex of S can be defined as:

$$\mathcal{D}(S) := \{\varsigma \subseteq S : \cap_{p \in \varsigma} \mathcal{V}(p, S) \neq \emptyset\}.$$

The following lemma is just another view of the definition of Delaunay complexes.

Lemma 1 (The empty ball property [22] (p. 48))**.** *Any subset $\sigma \subset S$ is a simplex of the Delaunay complex of S if and only if it has a circumscribing (open) ball empty of points of S.*

Given $d > 0$, an embedding of a simplicial complex K in the d-dimensional space $\mathbb{R}^d$ is usually called a geometric realization of K, and it will be denoted by $|K|$.

One of the key ideas along this paper is that a triangulation can be refined by successive subdivisions of the simplicial complex obtained from the triangulation. There are many different ways to obtain a subdivision of a simplex; in our case, we will use the barycentric subdivision.

Definition 7. *Let K be a simplicial complex with vertices in $\mathbb{R}^d$. The barycentric subdivision* Sd K *is the simplicial complex defined as follows. The set* $(\mathrm{Sd}\, K)^{(0)}$ *of vertices of* Sd K *is the set of barycenters of all the simplices of K. The simplices of* Sd K *are the finite nonempty collections of* $(\mathrm{Sd}\, K)^{(0)}$ *that are totally ordered by the face relation in K. That is, any k-simplex σ of* Sd K *can be written as an ordered set $\{w_0, \ldots, w_k\}$ such that w_i is the baricenter of μ_i, being μ_i a face of $\mu_j \in K$ for $i, j \in [\![0, k]\!]$ and $i < j$. In particular, if σ is maximal, then there exists a d-simplex $\{u_0, \ldots, u_d\} \in K$ satisfying that w_i is the barycenter of $\{u_0, \ldots, u_i\}$ for $i \in [\![0, d]\!]$.*

Let us introduce now the notion of simplicial approximation, which is a simplicial map defined on two simplicial complexes K and L that approximates a given continuous function g between the geometric realization of K and L. First, we recall the concept of vertex maps between two simplicial complexes.

Definition 8. *Given two simplicial complexes K and L, a vertex map $\varphi^{(0)} : K^{(0)} \to L^{(0)}$ is a function from the vertices of K to the vertices of L such that for any simplex $\sigma \in K$, the set*

$$\varphi(\sigma) := \{v \in L^{(0)} : \exists\, u \in \sigma,\ \varphi^{(0)}(u) = v\}$$

is a simplex of L.

A vertex map defined on the vertices of a simplicial complex K can be linearly extended to a continuous function on the whole simplicial complex K.

Definition 9. *The simplicial map $\varphi^c : |K| \to |L|$ induced by the vertex map $\varphi^{(0)} : K^{(0)} \to L^{(0)}$ is a continuous function defined as follows. Let $x \in |K|$. Then, $x \in |\sigma|$ for some simplex $\sigma = \{u_0, \ldots, u_k\}$ of K. So, $x = \sum_{i \in [\![0,k]\!]} \lambda_i u_i$ being $\lambda_i \geq 0$, for all $i \in [\![0, k]\!]$ and $\sum_{i \in [\![0,k]\!]} \lambda_i = 1$. Then,*

$$\varphi^c(x) := \sum_{i \in [\![0,k]\!]} \lambda_i \varphi^{(0)}(u_i).$$

Intuitively, a simplicial approximation between two simplicial complex K and L is a simplicial map that preserves the star of a vertex. Recall that for a vertex v of $K^{(0)}$, the star of v, denoted by $\operatorname{st} v$, is the set of simplices of K having $\{v\}$ as a face.

Definition 10. *Let $g : |K| \to |L|$ be a continuous function between the geometric realization of two simplicial complexes K and L. A simplicial map $\varphi^c : |K| \to |L|$ induced by a vertex map $\varphi^{(0)} : K^{(0)} \to L^{(0)}$ is a simplicial approximation of g if*

$$g(|\operatorname{st} v|) \subseteq |\operatorname{st} \varphi^c(v)|$$

for each vertex v of $K^{(0)}$.

Next, the main definition used in this paper is recalled. Given a simplicial map between the geometric realizations of two finite pure simplicial complexes, a two-hidden-layer feedforward network can be built. Such neural network is called a simplicial-map neural network and the value of its weights can be exactly computed from the vertex map associated to the simplicial map. In other words, there is no need to train the neural network to find the optimal weights.

Definition 11. *Let K and L be two finite pure simplicial complexes of dimension d and k, respectively. Let us consider the simplicial map $\varphi^c : |K| \to |L|$ induced by a vertex map $\varphi^{(0)} : K^{(0)} \to L^{(0)}$. Let $\{\sigma_1, \dots \sigma_n\}$ be the maximal simplices of K, where $\sigma_i = \{u_0^i, \dots, u_d^i\}$ and $u_h^i \in \mathbb{R}^d$ for $i \in [\![1, n]\!]$ and $h \in [\![0, d]\!]$. Let $\{\mu_1, \dots, \mu_m\}$ be the maximal simplices of L, where $\mu_j = \{v_0^j, \dots, v_k^j\}$ and $v_h^j \in \mathbb{R}^k$ for $j \in [\![1, m]\!]$ and $h \in [\![0, k]\!]$. The simplicial-map neural network induced by φ^c, denoted by $\mathcal{N}_\varphi$, is the two-hidden-layer feedforward neural network having the following architecture:*

- *an input layer with $d_0 = d$ neurons;*
- *a first hidden layer with $d_1 = n(d+1)$ neurons;*
- *a second hidden layer with $d_2 = m(k+1)$ neurons; and*
- *an output layer with $d_3 = k$ neurons.*

This way, $\mathcal{N}_\varphi = f_3 \circ f_2 \circ f_1$ being $f_i(y) = \phi_i(W^{(i)}; y; b_i)$, for $i \in [\![1, 3]\!]$, defined as follows.

First, $W^{(1)} = \begin{pmatrix} W_1^{(1)} \\ \vdots \\ W_n^{(1)} \end{pmatrix} \in \mathcal{M}_{n(d+1)\times d}$ and $b_1 = \begin{pmatrix} B_1 \\ \vdots \\ B_n \end{pmatrix} \in \mathbb{R}^{n(d+1)}$ where

$$\begin{pmatrix} W_i^{(1)} & | & B_i \end{pmatrix} = \begin{pmatrix} u_0^i & \cdots & u_d^i \\ 1 & \cdots & 1 \end{pmatrix}^{-1} \in \mathcal{M}_{(d+1)\times(d+1)}$$

being $W_i^{(1)} \in \mathcal{M}_{(d+1)\times d}$ and $B_i \in \mathbb{R}^{d+1}$. The function ϕ_1 is defined as

$$\phi_1(W^{(1)}; y; b_1) := W^{(1)} y + b_1.$$

Second, $W^{(2)} = \left(W_{h,\ell}^{(2)}\right) \in \mathcal{M}_{m(k+1)\times n(d+1)}$ $b_2 \in \mathbb{R}^{m(k+1)}$ is null where

$$W_{h,\ell}^{(2)} := \begin{cases} 1 & \text{if } \varphi^{(0)}(u_t^i) = v_r^j, \\ 0 & \text{otherwise;} \end{cases}$$

being $h = j(r+1)$ and $\ell = i(t+1)$ for $i \in [\![1, n]\!]$; $j \in [\![1, m]\!]$; $t \in [\![0, d]\!]$; and $r \in [\![0, k]\!]$. The function ϕ_2 is defined as:

$$\phi_2(W^{(2)}; y; b_2) := W^{(2)} y.$$

Thirdly, $W^{(3)} = \begin{pmatrix} W_1^{(3)} & \cdots & W_m^{(3)} \end{pmatrix} \in \mathcal{M}_{k\times m(k+1)}$ and $b_3 \in \mathbb{R}^k$ is null being

$$W_j^{(3)} := \begin{pmatrix} v_0^j & \cdots & v_k^j \end{pmatrix} \text{ for } j \in [\![1, m]\!].$$

The function ϕ_3 is defined as:

$$\phi_3(W^{(3)}; y; b_3) := \frac{\sum_{j\in[\![1,\ell]\!]} z^j \psi(y^j)}{\sum_{j\in[\![1,\ell]\!]} \psi(y^j)}$$

being $z^j := W_j^{(3)} y^j$ *for* $y = \begin{pmatrix} y^1 \\ \vdots \\ y^m \end{pmatrix} \in \mathcal{M}^{m\cdot(k+1)}$ *and*

$$\psi(y^j) := \begin{cases} 1 & \text{if all the coordinates of } y^j \text{ are} \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

As shown in [17], simplicial-map neural networks can be used for classification purposes. Given a labeled data set $D \subset \mathbb{R}^d \times \mathbb{R}^k$, we first compute a convex polytope $\mathcal{P}$ surrounding D. Second, we compute the Delaunay complex $K = \mathscr{D}(S)$ of the set $S = D_P \cup V_\mathcal{P}$ and define a simplicial complex L composed of a maximal simplex $\sigma = \{v_0, \ldots, v_\ell\}$ such that its dimension is equal to the number of classes of D. Finally, a vertex map that induces a simplicial-map neural network that correctly classifies D is defined as follows (see Proposition 4 in [17]):

$$\varphi^{(0)}(u) := \begin{cases} v_i & \text{if } (u,i) \in D, \\ v_0 & \text{if } u \in V_\mathcal{P}. \end{cases} \tag{1}$$

However, this simplicial-map neural network is not robust to adversarial attacks as shown in Proposition 5 in [17]. To construct simplicial-map neural networks robust to adversarial attacks of a given bounded size, the idea is to define a width decision boundary through barycentric subdivisions. Nevertheless, with each barycentric subdivision iteration, the number of simplices grows as it is claimed in Remark 1 of [16].

Once we have introduced all the necessary notions to explicitly construct a neural network to solve a classification problem, we present a methodology to reduce the size of such a network without hindering its performance.

3. Description of the Methodology

In this section, we propose a methodology to reduce the size of a simplicial-map neural network used for classification tasks.

Recall that given a labeled data set D with k classes, the process to obtain a simplicial-map neural network that correctly classifies D is: (1) to compute a convex polytope $\mathcal{P}$ surrounding D; (2) to compute the Delaunay complex K of the set $D_P \cup V_\mathcal{P}$; (3) to compute a vertex map $\varphi^{(0)}$ from the vertices of K to the vertices of a simplicial complex L with only one maximal k-simplex; and (4) to compute a simplicial-map neural network $\mathcal{N}_\varphi : |K| \to |L|$, from the simplicial map φ^c.

However, this simplicial-map neural network $\mathcal{N}_\varphi$, as many other neural networks, can suffer the attack of adversarial examples. In [17], a method to increase the robustness of the simplicial-map neural network to such attacks was developed by applying successive barycentric subdivisions to K and L depending on the desired robustness. However, the iteration of barycentric subdivisions results in the exponential growth of the number of simplices. Therefore, the storage and computational cost of the simplicial map φ^c and the simplicial-map neural network $\mathcal{N}_\varphi$ grow exponentially.

In order to avoid this problem, in this paper, we propose a method to reduce the storage and computational cost of the simplicial-map neural network $\mathcal{N}_\varphi : |K| \to |L|$ by removing points of the given labeled data set D but keeping exactly the same accuracy as $\mathcal{N}_\varphi$. The idea is to remove those simplices from K whose vertices belong all to the same class. Therefore, those simplices with vertices in the decision boundary remain, leaving the decision boundary invariant.

Let us now formalize this idea. Let $D = \{(p_j, \ell_j) : j \in [\![1, n]\!],\ p_j \in \mathbb{R}^d,\ \ell_j \in \mathbb{E}^k\}$ be a data set and let $\mathcal{N}_\varphi$ be the simplicial-map neural network obtained using the process described above. Our aim is to obtain a subset $\tilde{D}$ that induces a simplicial-map neural network $\tilde{\mathcal{N}}_{\tilde{\varphi}}$ with exactly the same behavior than $\mathcal{N}_\varphi$. The procedure is described in Algorithm 1.

Algorithm 1 Simplicial-map neural network optimization

Result: The simplicial-map neural network $\tilde{\mathcal{N}}_{\tilde{\varphi}}$;
Input: A labeled data set D, a convex polytope $\mathcal{P}$ surrounding D, and the simplicial-map neural network $\mathcal{N}_\varphi : |K| \to |L|$ that correctly classifies D;
$M := \emptyset$;
for $\sigma = \{v_0, \ldots, v_n\} \subset D_\mathcal{P} \cup V_\mathcal{P}$ *being a maximal simplex of* K **do**
  if $\mathcal{N}_\varphi(v_i) \neq \mathcal{N}_\varphi(v_j)$ *for some* $i \neq j$ **then**
    $M := M \cup \{\sigma\}$;
  end
end
$\tilde{D} := \{(v, \ell) : v \in M^{(0)}$ and $(v, \ell) \in D\}$;
$\tilde{K} := \mathscr{D}(\tilde{D}_\mathcal{P} \cup V_\mathcal{P})$;
Compute the simplicial map $\tilde{\varphi}^{(0)} : \tilde{K}^{(0)} \to L^{(0)}$;
Compute the simplicial-map neural network $\tilde{\mathcal{N}}_{\tilde{\varphi}} : |\tilde{K}| \to |L|$;
Output: $\tilde{\mathcal{N}}_{\tilde{\varphi}}$

In Section 4, using a high-dimensional data set composed of digit images, we check experimentally that both simplicial-map neural networks $\tilde{\mathcal{N}}_{\tilde{\varphi}}$ and $\mathcal{N}_\varphi$ have the same behavior. The following partial result also supports that idea.

Lemma 2. *Let D be a labeled data set, let $\mathcal{N}_\varphi : |K| \to |L|$ be the simplicial-map neural network that correctly classifies D, constructed following the method given in [17], and let $\tilde{\mathcal{N}}_{\tilde{\varphi}}$ be the simplicial-map neural network obtained from Algorithm 1. If $\sigma = \{v_0, \ldots, v_n\} \in K$ satisfies that $\mathcal{N}_\varphi(v_i) \neq \mathcal{N}_\varphi(v_j)$ for some $i \neq j$, then $\tilde{\mathcal{N}}_{\tilde{\varphi}}(x) = \mathcal{N}_\varphi(x)$ for all $x \in |\sigma|$.*

Proof. Let $\sigma = \{v_0, \ldots, v_n\}$ be a simplex of K such that $\mathcal{N}_\varphi(v_i) \neq \mathcal{N}_\varphi(v_j)$ for some $i \neq j$. Then, σ is a face of a maximal simplex μ of K with all its vertices belonging to $\tilde{D}_\mathcal{P} \cup V_\mathcal{P}$. Therefore, μ is a maximal simplex of $\tilde{K}$ (by Lemma 1) and $\tilde{\mathcal{N}}_{\tilde{\varphi}}(x) = \mathcal{N}_\varphi(x)$ for any $x \in |\mu|$. Since σ is a face of μ then $\tilde{\mathcal{N}}_{\tilde{\varphi}}(x) = \mathcal{N}_\varphi(x)$ for any $x \in |\sigma|$. □

In order to illustrate Algorithm 1, let us consider the two-dimensional labeled data set D given in Figure 1. Let us consider a square surrounding the data set as the convex polytope $\mathcal{P}$, and let us compute the Delaunay complex $K = \mathscr{D}(D_\mathcal{P} \cup V_\mathcal{P})$ as shown in Figure 2. Then, K is composed of 24 points and 42 2-simplices. Applying Algorithm 1 is equivalent to remove those 2-simplices of K whose vertices belong, all of them, to the same class. Then, we consider only the vertices of the surviving 2-simplices and the Delaunay complex is computed again. In that case, the resultant simplicial complex is composed of 18 points and 30 2-simplices (see Figure 2).

Lemma 3. *If the points of $D_\mathcal{P} \cup V_\mathcal{P}$ are in general position, then the reduced simplicial neural network $\tilde{\mathcal{N}}_{\tilde{\varphi}}$ can always be computed from Algorithm 1.*

Proof. If the points of $D_\mathcal{P} \cup V_\mathcal{P}$ are in general position, then any subset of points of $D_\mathcal{P} \cup V_\mathcal{P}$ are in general position, so the the Delaunay triangulation of $\tilde{D}_\mathcal{P} \cup V_\mathcal{P}$ can always be computed, as well as the simplicial-map neural network $\tilde{\mathcal{N}}_{\tilde{\varphi}}$. □

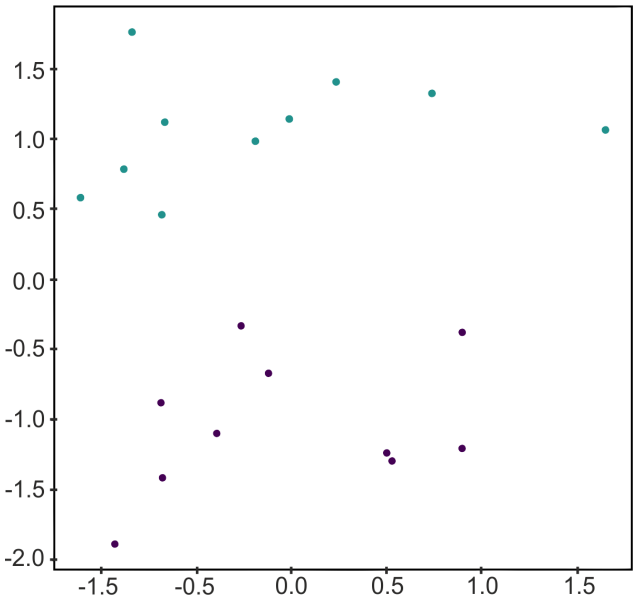

Figure 1. A toy example of a two-dimensional data set for binary classification generated using the scikit-learn package implementation of [23].

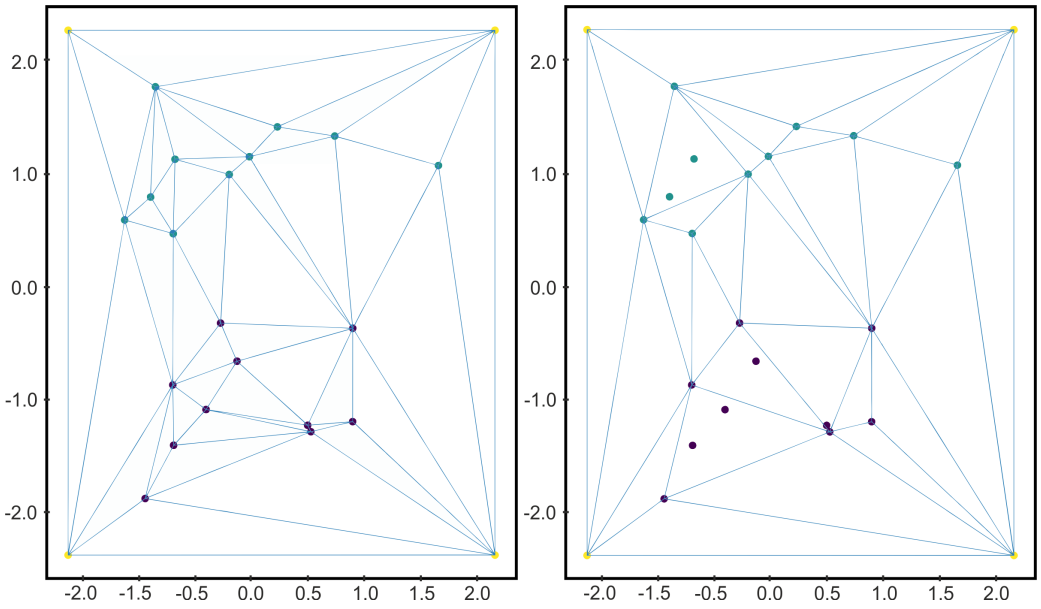

Figure 2. On the left, the Delaunay triangulation of the labeled data set D given in Figure 1 together with the vertices of the square polytope surrounding D. On the right, the Delaunay triangulation of a subset of D obtained as described in Algoritm 1. As we can see, the triangles whose vertices belonged all to the same class disappeared.

Let us notice that, depending on the distribution of the data set, the reduction obtained after applying Algorithm 1 can be significant or not. Specifically, if the different classes of D are not mixed, then we can expect good results of Algorithm 1. The reduction will be optimum when the data set is separable and dense. In such case, most of the simplices would have vertices of the same class and be removed when Algorithm 1 is applied. An example of these two opposite cases are shown in Figure 3.

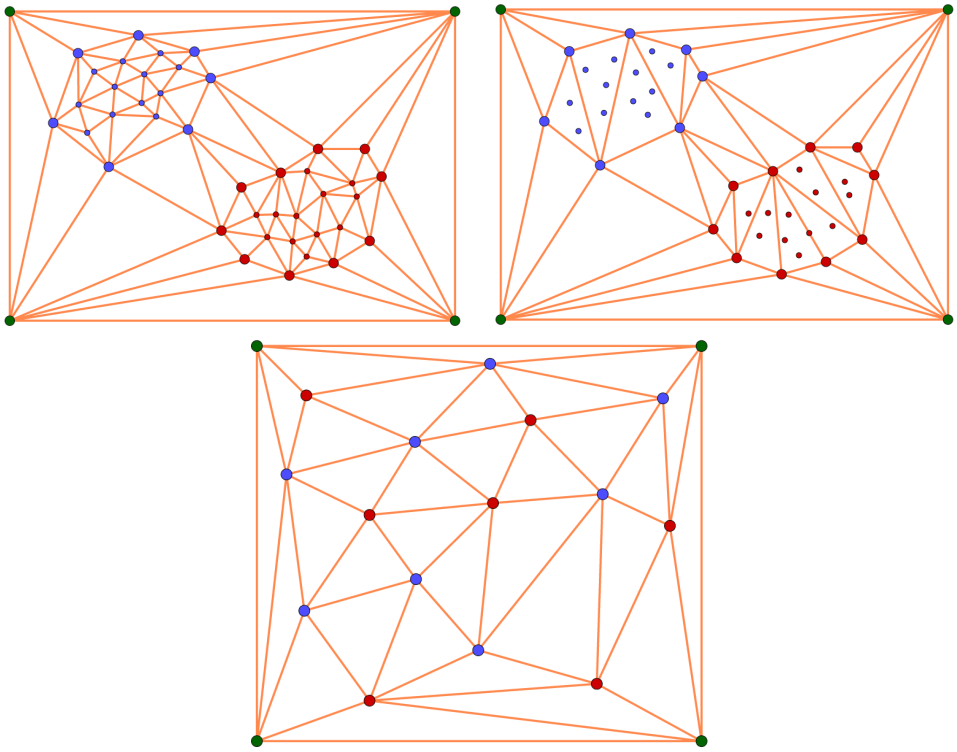

Figure 3. From left to right and from top to bottom: (1) a separable and dense binary data set D; (2) the data set obtained after applying Algorithm 1 to D; (3) a data set that cannot be reduced by applying Algorithm 1.

4. Experiments

In this section, a high-dimensional data set composed of digit images is considered. In this case, for visualization purposes, the data set is firstly embedded to obtain a low-dimensional representation using the UMAP algorithm [24]. The data set is composed of 1797 greyscale images of dimension 8×8. These images represent digits from 0 to 9. In Figure 4, some of the images are shown and, in Figure 5, the two-dimensional UMAP output is displayed, representing the full data set. In order to illustrate our method by providing a graphical intuition, we will focus on the 2D representation of the digits data set, but the construction can be conducted with any dimension of the input.

Let us focus on the 1797 two-dimensional points of the UMAP representation of the digits data set D depicted in Figure 5, and let us consider a square $\mathcal{P}$ surrounding such a cloud of points D_P. According to [17], a simplicial-map neural network $\mathcal{N}_\varphi$ can be built in order to correctly classify D. Now, let us apply Algorithm 1 to obtain a simplified version of $\mathcal{N}_\varphi$ that also correctly classify D. This way, all of the points in D_P surrounded by points belonging to the same class were removed to obtain a reduced data set $\tilde{D}$ inducing the same simplicial-map neural network than D. In Figure 5, the two-dimensional representation of the reduced data set is shown. The next step is the computation of the Delaunay triangulation using the data set $\tilde{D}$ and the vertices of the square $\mathcal{P}$. In Figure 6, the Delaunay triangulation is shown for both the original and the simplified data set. The Delaunay triangulation of the original data set is composed of 3596 2-simplices, whereas the Delaunay triangulation of the simplified data set is composed of 604 2-simplices and 305 points reaching a remarkable reduction in the number of simplices. The results are summarized in Table 1. Finally, the induced simplicial-map neural networks were experimentally compared obtaining exactly the same performance.

Lastly, Algorithm 1 was experimentally tested for synthetically generated two- and three-dimensional data sets. The numerical results can be found in Tables 2 and 3, respectively. Let us point out that in the three-dimensional data set with a greater amount of points,

the reduced data set has a reduction of approximately 73%, inducing the same simplicial-map neural network.

The code of the experimentation can be consulted in https://github.com/Cimagroup/DelaunayTriangAndNN (accessed on 30 August 2021).

Figure 4. Some of the 1797 images used in the experiment. The images are labeled from 0 to 9 in a natural way. Each image is grey-scaled and has 8×8 pixels, so it can be represented as a point in $\mathbb{R}^{64}$. In order to visualize such 1797 64-dimensional points, $\mathbb{R}^{64}$ has been projected into $\mathbb{R}^2$ using the UMAP algorithm. Figure 5 shows the projection on $\mathbb{R}^2$ of the 1797 images.

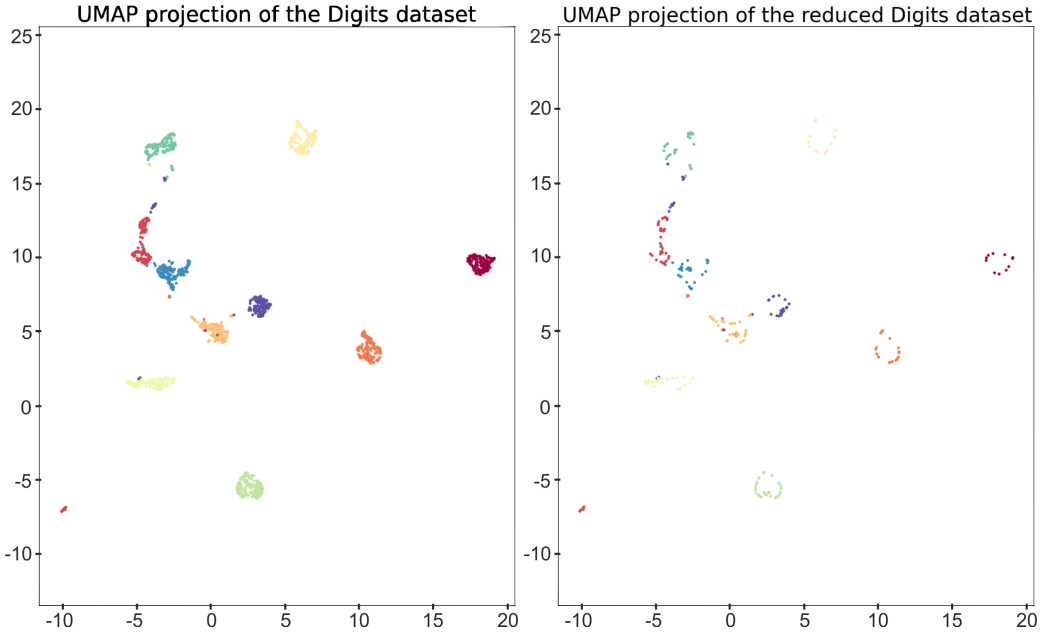

Figure 5. Visualization of the UMAP 2D representation of the original data set used (**left**), and the simplified data set obtained (**right**).

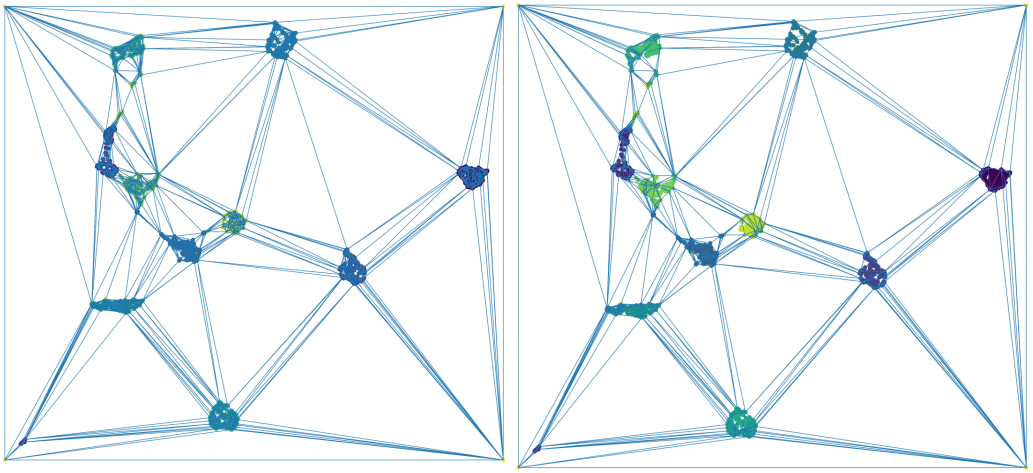

Figure 6. On the (**left**), the Delaunay triangulation of the original data set and, on the (**right**), the Delaunay triangulation of the simplified data set.

Table 1. The size of the data set used in the digits experiment, the number of 2-simplices obtained after computing the Delaunay triangulation, and the ones obtained after applying Algorithm 1.

Data Set Size	2-Simplices	2-Simplices (Reduced)	Data Set Size (Reduced)
1801	3596	604	305

Table 2. The size of the two-dimensional synthetic data sets used, the number of 2-simplices obtained after computing the Delaunay triangulations, and the ones obtained after applying Algorithm 1.

Data Set Size	2-Simplices	2-Simplices (Reduced)	Data Set Size (Reduced)
14	22	22	14
104	202	58	32
1004	2002	230	118
10,004	20,002	8384	4195
100,004	200,002	6620	3313
1,000,004	2,000,002	73,488	36,747

Table 3. The size of the three-dimensional synthetic data sets used, the number of 2-simplices obtained after computing the Delaunay triangulations, and the ones obtained after applying Algorithm 1.

Data Set Size	3-Simplices	3-Simplices (Reduced)	Data Set Size (Reduced)
14	34	29	13
104	551	391	75
1004	6331	1647	272
10,004	66,874	30,357	4556
100,004	672,097	147,029	21,955
1,000,004	6,762,603	1,858,204	274,635

5. Conclusions

Simplicial-map neural networks are a recent neural network architecture based on simplicial maps defined between a triangulation of the given data set and a simplicial complex encoding the classification problem. These neural networks are refined by applying barycentric subdivisions to ensure their robustness. The iterative application of barycentric subdivisions increases the number of simplices exponentially. Therefore, the width of the neural network also increases exponentially. In this paper, we have provided a way to reduce the number of simplices but maintaining the performance of the neural network. The proposed method has been experimentally tested. As further work, we plan to formally prove that our optimized simplicial-map neural network $\tilde{\mathcal{N}}_{\tilde{\varphi}}$ is equivalent to the original one $\mathcal{N}_{\varphi}$.

Author Contributions: Conceptualization, R.G.-D., M.A.G.-N., J.H. and E.P.-H.; methodology, R.G.-D., M.A.G.-N., J.H. and E.P.-H.; formal analysis, R.G.-D., M.A.G.-N., J.H. and E.P.-H.; investigation, R.G.-D., M.A.G.-N., J.H. and E.P.-H.; writing—original draft preparation, R.G.-D., M.A.G.-N., J.H. and E.P.-H.; writing—review and editing, R.G.-D., M.A.G.-N., J.H. and E.P.-H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Agencia Estatal de Investigación under grant PID2019-107339GB-100.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code and data of the experimentation can be consulted in https://github.com/Cimagroup/DelaunayTriangAndNN.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org (accessed on 30 August 2021).
2. Zhai, X.; Kolesnikov, A.; Houlsby, N.; Beyer, L. Scaling Vision Transformers. *arXiv* **2021**, arXiv:2106.04560.
3. Yu, G.X.; Gao, Y.; Golikov, P.; Pekhimenko, G. Computational Performance Predictions for Deep Neural Network Training: A Runtime-Based Approach. *arXiv* **2021**, arXiv:2102.00527.
4. Strubell, E.; Ganesh, A.; McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. *arXiv* **2019**, arXiv:1906.02243.
5. Gonzalez-Diaz, R.; Paluzo-Hidalgo, E.; Gutiérrez-Naranjo, M.A. Representative datasets for neural networks. *Electron. Notes Discret. Math.* **2018**, *68*, 89–94. [CrossRef]
6. Sener, O.; Savarese, S. Active Learning for Convolutional Neural Networks: A Core-Set Approach. *arXiv* **2018**, arXiv:1708.00489.
7. Nguyen, A.T.; Andreis, B.; Lee, J.; Yang, E.; Hwang, S.J. Stochastic Subset Selection. *arXiv* **2021**, arXiv:2006.14222.
8. Sucholutsky, I.; Schonlau, M. Soft-Label Dataset Distillation and Text Dataset Distillation. *arXiv* **2020**, arXiv:1910.02551.
9. Reed, R. Pruning algorithms-a survey. *IEEE Trans. Neural Netw.* **1993**, *4*, 740–747. [CrossRef] [PubMed]
10. Frankle, J.; Dziugaite, G.K.; Roy, D.M.; Carbin, M. Pruning Neural Networks at Initialization: Why are We Missing the Mark? *arXiv* **2020**, arXiv:2009.08576.
11. Xu, D.; Yen, I.E.H.; Zhao, J.; Xiao, Z. Rethinking Network Pruning—Under the Pre-train and Fine-tune Paradigm. *arXiv* **2021**, arXiv:2104.08682.
12. Lazarevich, I.; Kozlov, A.; Malinin, N. Post-training deep neural network pruning via layer-wise calibration. *arXiv* **2021**, arXiv:2104.15023.
13. Gale, T.; Elsen, E.; Hooker, S. The State of Sparsity in Deep Neural Networks. *arXiv* **2019**, arXiv:1902.09574.
14. Han, S.; Pool, J.; Tran, J.; Dally, W.J. Learning both Weights and Connections for Efficient Neural Networks. *arXiv* **2015**, arXiv:1506.02626.
15. Frankle, J.; Schwab, D.J.; Morcos, A.S. The Early Phase of Neural Network Training. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
16. Paluzo-Hidalgo, E.; Gonzalez-Diaz, R.; Gutiérrez-Naranjo, M.A. Two-hidden-layer feed-forward networks are universal approximators: A constructive approach. *Neural Netw.* **2020**, *131*, 29–36. [CrossRef] [PubMed]
17. Paluzo-Hidalgo, E.; Gonzalez-Diaz, R.; Gutiérrez-Naranjo, M.A.; Heras, J. Simplicial-Map Neural Networks Robust to Adversarial Examples. *Mathematics* **2021**, *9*, 169. [CrossRef]
18. Wang, Y.; Jha, S.; Chaudhuri, K. Analyzing the Robustness of Nearest Neighbors to Adversarial Examples. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 5133–5142.
19. Langenberg, P.; Balda, E.; Behboodi, A.; Mathar, R. On the Robustness of Support Vector Machines against Adversarial Examples. In Proceedings of the 2019 13th International Conference on Signal Processing and Communication Systems (ICSPCS), Gold Coast, Australia, 13–15 December 2019; pp. 1–6. [CrossRef]
20. Hornik, K. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Netw.* **1991**, *4*, 251–257. [CrossRef]
21. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
22. Boissonnat, J.D.; Chazal, F.; Yvinec, M. *Geometric and Topological Inference*; Cambridge Texts in Applied Mathematics; Cambridge University Press: Cambridge, UK, 2018. [CrossRef]
23. Guyon, I.; Gunn, S.; Hur, A.B.; Dror, G. Design and Analysis of the NIPS2003 Challenge. In *Feature Extraction: Foundations and Applications*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 237–263. [CrossRef]
24. McInnes, L.; Healy, J.; Melville, J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv* **2020**, arXiv:1802.03426.