Iraqi Journal of Science, 2022, Vol. 63, No. 5, pp: 2262-2278
DOI: 10.24996/ijs.2022.63.5.37

ISSN: 0067-2904

Using Persistence Barcode to Show the Impact of Data Complexity on the Neural Network Architecture

Labiba M. Alhelfi [*], Hana M. Ali
Department of Mathematics, College of Science, University of Basrah, Basrah, Iraq

Received: 5/7/2021 Accepted: 19/9/2021 Published: 30/5/2022

Abstract
It is so much noticeable that initialization of architectural parameters has a great impact on whole learnability stream so that knowing mathematical properties of dataset results in providing neural network architecture a better expressivity and capacity. In this paper, five random samples of the Volve field dataset were taken. Then a training set was specified and the persistent homology of the dataset was calculated to show impact of data complexity on selection of multilayer perceptron regressor (MLPR) architecture. By using the proposed method that provides a well-rounded strategy to compute data complexity. Our method is a compound algorithm composed of the t-SNE method, alpha-complexity algorithm, and a persistence barcode reading method to extract the Betti number of a dataset. After that, MLPR were trained using that dataset using a single hidden layer with increased hidden neurons. Then, increased both hidden layers and hidden neurons. Our empirical analysis has shown that the training efficiency of MLPR severely depends on its architecture's ability to express the homology of the dataset.

Keywords: ANN Architecture, Alpha Complexity, Topological Data Analysis, Betti Number, Persistence Barcode.

إستِخدامُ الباركود المستمر لإظهارِ تأثيرِ تعقيدِ البياناتِ على بنيةِ الشبكةِ العصبية

لبيبة مضير عبد الوهاب*, هناء مرتضى علي
قِسمُ الرياضيات, كُليةُ العلوم, جامعةُ البَصرة, البَصرة, العِراق

ألخُلاصة
من الملاحظ إلى حد كبير أن تهيئة المعلمات الهيكلية لها تأثير كبير على مجرى قابلية التعلم بالكامل ، بحيث تؤدي معرفة الخصائص الرياضية لمجموعة البيانات إلى توفير بنية الشبكة العصبية بتعبير وقدرة أفضل. في هذا البحث ، تم أخذ خمس عينات عشوائية من مجموعة بيانات حقل Volve ، ثم تم تحديد مجموعة التدريب وحُسب التماثل المستمر لمجموعة البيانات لإظهار تأثير تعقيد البيانات على اختيار بنية multilayer perceptron regressor (MLPR) باستخدام الطريقة المقترحة التي توفر استراتيجية شاملة لحساب تعقيد البيانات. طريقتنا عبارة عن خوارزمية مركبة تتكون من طريقة t-SNE وخوارزمية تعقيد ألفا وطريقة قراءة الباركود لاستخراج Betti number من مجموعة البيانات. بعد ذلك ، تم تدريب MLPR

*Email: dhmary033@gmail.com

باستخدام مجموعة البيانات تلك باستخدام طبقة مخفية مفردة مع زيادة الخلايا العصبية مخفية . بعد ذلك ، زادت الطبقات المخفية والخلايا العصبية المخفية. أظهر تحليلنا التجريبي أن كفاءة تدريب MLPR تعتمد بشدة على قدرة بنيتها على التعبير عن التماثل في مجموعة بيانات التدريب.

1. Introduction

In the past century, due to rapid progress in technological ideas and their applications, it was no longer possible to exploit some of mathematical models, especially statistical models in huge data-based methods, because it requires high computing power. Thus providing a computational technique that helps to reduce the time complexity and elevating the computation power of learning algorithms was severely required to push the science forward by cloud computing that decreases the training time of learning algorithms especially in deep learning.

Deep learning is a subfield of machine learning consisting of a set of learning-based models that mimics the brain's functioning and structure to train a computer system for achieving results by using pre-available data [1]. Deep learning is constantly a rising trend in the data analysis field, has been labeled one of the 10 advanced strategies of 2013 [2-4]. Substantially, the most common technique in deep learning is ANNs that have been adjusted with more layers to provide multiple levels of abstractions and better data predictions [2]. Regarding this matter, A deep neural network (DNN), which is a representation-learning method comprised of multiple layers, is supplied with a huge amount of raw data to discover the optimal data representation that can be used for prediction or classification [5].

Starting from the raw data, each non-linear unit embedded in DNN transforms the representation of data at one level into a higher level of abstraction. Specifically, a non-linear transformation on the output vector of the previous layer is performed by the next layer, thereby the representation of data is continuously exposed to several transformations as the data is permanently and successively being forwarded from one layer to another. The non-linear transformation modules that involve in such alteration of data representation are the weight matrix_between two consecutive layers, biases, and the activation functions [6]. To obtain the optimal representation of data in a backpropagation network, the ANN trains itself by leaning on a gradient-based optimization procedure by which the decision variables of the weight vector are being adjusted to minimize the objective function of training error in each training epoch. At the end of the final epoch, the model's capacity should be perfectly optimized in such a way that avoids the occurrence of a high generalization gap between the training and the testing errors (overfitting) [7]. Consequently, the key sight in the utilization of the adjustable variables of weights and biases along with activation functions is to reach the point at which the model would have a better representation of data to interact with. Accordingly, the extraction of such a high-level representation enables the model to acquire an outstanding capability for splitting data as in neural network classifier model or fitting in case of neural network regressor.

As illustrated above, the adequate technique to detect an optimal capacity of DNN can be achieved by selecting the appropriate hypothesis space of the hidden units (list of activation functions) that fits well with the amount and distribution of examples [7]. Nevertheless, a question may arise on whether the detection of the optimal number of hidden layers (depth) and hidden neurons (width) would enhance the model capacity. Undoubtedly, the significant challenge in each domain is establishing the model architecture that takes advantage of the structure in data optimally [8].

In ANN, since hidden neurons and layers are the main units, initially, determination of ANN architecture, depth and width, is the most important issue that should be considered before initialization of any other hyperparameters [9, 10]. Furthermore, during the training process, majority of other hyperparameters can be modified accurately with less computational complexity if the architecture of ANN were initialized ideally. Both feedforward and backpropagation are affected by early state of ANN architectural design. During feeding of data, each hidden layer applies geometric transformations which are a linear transformation implemented by the matrix of weights W, a translation implemented by the bias vector b, and utilization of activation functions which provides two advantages, one for a restriction of the new representation's magnitude to a certain limit, the other is to provide non-linearity to the model. On the other hand, a better convergence with less computation can be performed by an optimization algorithm, if the size of the weights matrix where scrupulously determined. Thus, If the optimal selection of the model's representational capacity increases the model performance, then the size of depth and width do likewise.

On the other hand, it was proved that increasing width and depth of ANN is the most straightforward method to improve their performance; however, this easy approach has two main pitfalls clarified in [11]. Mainly, a larger size usually means a larger number of parameters, which makes the expanded network more vulnerable to overfitting, particularly if the training set has a limited number of labeled examples. Besides this capacity-related problem, from a computation perspective, more adjustable parameters, more dramatic increase of running time. By taking such drawbacks as a challenge. Therefore, overall idea was to overcome the problem of architectural parameters selection of the ANNs to improve their capacity and rein the computation complexity of the training process as low as reasonably achievable.

Since ANN are learning from examples type of model, undoubtedly, the properties of the training dataset are main influencing factor of ANNs learning performance. Thus, having prior knowledge of raw data properties such as its probability distribution behavior, correlations, and other statistical features significantly helps in determining the most generalizable and expressive architecture that would be suitable for that data. In addition to those properties, topological features of a dataset can also be considered as a helpful guider concerning this matter and were therefore utilized in this paper as a determining factor for the optimal and less-computational architecture of ANN. In this paper, we employed a topological data analysis technique (TDA) to detect and show the relationship between the optimal architecture of MLPR and the topological complexity (TC) of the training dataset itself. Our technique has merged a dimensionality reduction method called t-SNE proposed by [12]. We preferred to take advantage of t-SNE since the authors in [13] reported that t-SNE can perform well with various data types, along with a filtration algorithm called alpha complexity and a persistent barcode reading method to extract the Bitti number (BN) of datasets, and thus detecting the topological complexity of a dataset.

1. Related Work

In Deep learning, the essential features of hierarchical representation and optimization have revolutionized the state of the art [7]. Enhancing generalization capacity and reducing computation complexity of neural networks have been targeted profusely, particularly through improving preliminary estimate of the structural parameters of ANN [14]. Many works have been concentrated on enhancing the expressivity and capacity of ANNs by trying to attain their perfect width and depth as we have discussed in a few next lines.

Pruning is one of the popular approaches to obtain optimal network architecture. For instance, an algorithm called pruning by significance (N2PS) which was proposed in [15] estimates the significance value of neurons, thereby prunes the unimportant neurons. This pruning algorithm is implemented by estimating the significance value of each input neuron hidden neuron in trained neural network and assign significance values to those neurons. Such estimated values were calculated by the Sigmoidal activation value of a neuron along with all weights of its outgoing connections. Thus, N2PS prunes all superfluous neurons which have significance values less than an estimated threshold.

Swarm intelligence algorithms have been also utilized in optimal architecture detection issues. In reference [16], the authors implemented the particle swarm optimization algorithm (PSO) to detect optimal architecture of recurrent neural networks. In that PSO, each particle in initial population consists of three parameters which were learning rate, number of hidden neurons, and activation function. Consequently, their goal was to find the best particle that has the optimal collection of pre-mentioned parameters including the optimal number of hidden neurons. employing the particle swarm for converging at the optimal network's architecture can be considered as an accurate approach. However, from a computation perspective, this method is still costly, since each particle represents a neural network itself that needs to be executed when calculating the fitness function of particles.

Since they have been raised, convolutional neural networks (CNN) own the lion's share regarding architectural issues. The authors in [11] focused on enhancing the initial architectural choices of the deep CNN for image classification proposed by [17]. They utilized Hebbian principle and intuition of multi-scale processing to design the optimal architectural decisions that aim to cautiously increase width and depth of the network while maintaining the computational budget constant. On the other hand, neural architecture search (NAS) in [18] adopted compositional hyperparameter search as a selection mechanism of neural architecture through which rather than attempting to find a single best architecture, they suggested using the concept of a "fabric" which is a system that contains an exponentially huge amount of architectures. in addition to the recovering of individual architectures as paths, the fiber will gather all embedded architectures together, exchanging their weights where their paths overlap.

In [19], by comparing between deep multilayered perceptrons (MLPs) and shallow ones in terms of the number of linearity regions, the authors proposed first step towards providing a mathematical analysis tool to discover the behavior of rectifier MLPs by attempting to study the relationship of MLPs depth and width to their complexity of sublevel sets. This was a motivation for [20] to propose a new metric, based on the BN, to assess the complexity of functions performed by ANN. This metric was utilized to show that deep networks can realize maps with more complexity than the shallows.

The authors in [21] clarified how the topological assumptions on input data result in efficiently expressive architectures. For this purpose, they proposed a novel layers parameterization framework called deep function machines (DFMs). DFMs are mathematical methods that can be used to produce provable properties in ANNs. They showed a great result in finding a better architecture of RippLeNET by utilizing the expressive ability of their DFMs. In [22], the authors proposed a mathematical tool to explore and understand the neural networks' nonlinear maps. This tool which was based on Riemannian geometry and dynamical mean-field theory is used to provide basic reconnoitering of deep neural networks expressivity.

In [23], the authors proposed a TDA technique that aims to show the impact of data complexity on the architecture expressivity of CNN. Their technique was leaned on Vietoris-Rips complex which is a computation costly filtration method. Besides, they have utilized two-dimensional persistent barcodes to compute the BN of their dataset. They have shown ability of CNN architectures to express various homology of datasets. They have shown an outstanding result to determine the width and depth of CNN. However, unfortunately, They did not clarify their method for extracting the BN, or how their algorithm behaves when utilizing a huge amount of dataset.

Since ANNs are platforms for calculating the nonlinear maps across high-dimensional spaces which must be compatible with the geometrical nature of the input dataset. From an algebraic topology perspective, The topological nature of datasets can be detected by exploring their complexity. Therefore, Efficient learning ability of neural network architectures can be subject to complexity of datasets as we concluded later in the next sections. As a result, in this paper, we proposed a composite low-cost TDA algorithm to compute data complexity. Our algorithm is based on a dimensionality reduction called t-SNE, alpha complexity filtration algorithm, and a persistent barcode reading method for extracting the BN. Through utilizing this algorithm, we have shown effects of the data homology on the expressivity of the depth and width of the MLPR model. Our overall goal is to establish a practical link between the preceding theories on ANNs expressive ability and data homology, with potential ability to reduce search space of ANNs architectures.

3. Background

Modern algebraic topology has emerged to offer quantitative methods for discovering and analyzing the "shapes" of geometric objects without any need for distances. It appoints algebraic invariants to geometric objects based on the relative position of points [24]. Historically, basic theory of algebraic topology has been associated with characterizing topological spaces based on their global properties by utilizing the algebraic invariants [25]. Nevertheless, theories from algebraic topology have recently been extended to the field of datasets as a means of analyzing the "shapes" of data by performing specialized algorithms. Topological spaces can be classified depending on the numbers and categories of holes embedded in those spaces [26]. For instance, a circle differs from a disk due to the existence of a hole in its middle. One of the core advantages of algebraic topology is providing powerful mechanisms to compare the complexity of shapes, which can be carried out by distinguishing among the topological spaces, whether it is topologically equivalent (or distinct), by assigning them to algebraic objects (e.g: chains and groups) inorder to finding out whether those objects are isomorphic (or not) [27]. Although there are several sorts of algebraic topology methods that deal with measuring complexity, homology is an efficient and computationally feasible method to compute the complexity of the topological spaces [28]. The following described the definition of homology group.

Definition 1.1 suppose X as a topological space with n-dimension holes that has a number denoted by β_n, then $H_n(X) = \mathbb{Z}^{\beta_n}$ is defined as the n^{th} **homology group**. Bear in mind that β_0 is the number of separate connected components (clusters); β_1 represents circle; β_2 are the gaps, and $\beta_n(X)$ is called nth **Betti number** of X [23].

Definition 1.2 if X is topological space and $H_n(X)$ is n^{th}homology group of X, then **the homology** $H(X) = \{H_n(X)\}_0^{\infty}$ [23].

Example 1.1 Let D_1 and D_2 are datasets, we assume that D_1 is the union of two solid balls, D_2 is the union of two solid balls and two rings. Since there are two separated clusters (0-dimensional holes) in D_1, $H_0(D_1) = \mathbb{Z}^2$, and since there is no ring (one-dimensional holes), then $H_1(D_1) = \{0\}$. Hence, $H_n(D_1) = \{0\}$ for $n \geq 2$. For D_2, since there are four separated clusters, $H_0(D_2) = \mathbb{Z}^4$, and since there are two rings, then $H_1(D_2) = \mathbb{Z}^2$. As a result, D_1 is less complex than D_2. Note that, β_0 is the number of 0-dimensional holes (clusters), β_1 is the number of one-dimensional holes (circles), and β_2 is the number of two-dimensional holes (gaps or cavities).

Corollary 1.1. if $\psi: X \to Y$ is a homeomorphism map then $\psi^*: H_n(X) \to H_n(Y)$ is an isomorphism [23].

During the homology's determination in the previous example, we were compelled to presume that our data (or spaces) were in form of geometric shapes. But, to properly take advantage of homology, we have to be able to calculate the homology directly on the data with the continuity of capturing significative topological features (e.g: clusters, circles, and gaps). For doing so, here comes the ability of persistent homology, which is a topological data analysis method, to recognize the valuable features in the spaces at multiple levels. In general, homology helps to calculate the number of n-dimensional holes in the space, and persistent homology provides a method to track those features as the scale grows [29]. For better understanding, we point the reader to great expository papers on persistent homology in [30] and [31]. Persistent homology defines the topological features of the data by converting data into simplicial complexes by utilizing a filtration method. Particularly, filtration is a topological space X having a sequence of subspaces $X = X_0 \subset X_1 \subset \cdots \subset X_n = X'$ defined as filtered simplicial complex.

The persistent homology is computed as $\psi_n{}^m: H_m(X) \to H_m(X')$ on the simplicial complex. Furthermore, the final result can be stored in various visualization structures such as a multi-set structure called *persistence diagram* or in *persistence barcode* [32]. The barcode indicates the topological features that are computed from a simplicial complex [33]. In Figure 1, which represents the barcode of 3D topological space, the red, green, and purple bars represent β_0, β_1,and β_2 respectively. The left endpoint of bars denotes the birth of betti (β_n) that refers to the time at which the homology discovers a specific feature, and the right endpoint denotes the death which refers to the time at which the feature becomes unrecognizable in the filtration. A bar with a short distance refers to a feature with a short life which is considered as a *topological noise*. On contrary, a long-life feature is considered a *topological feature* of the given space [34]. These diagrams will be utilized frequently for computing the persistent homology of datasets later on in the proposed work.

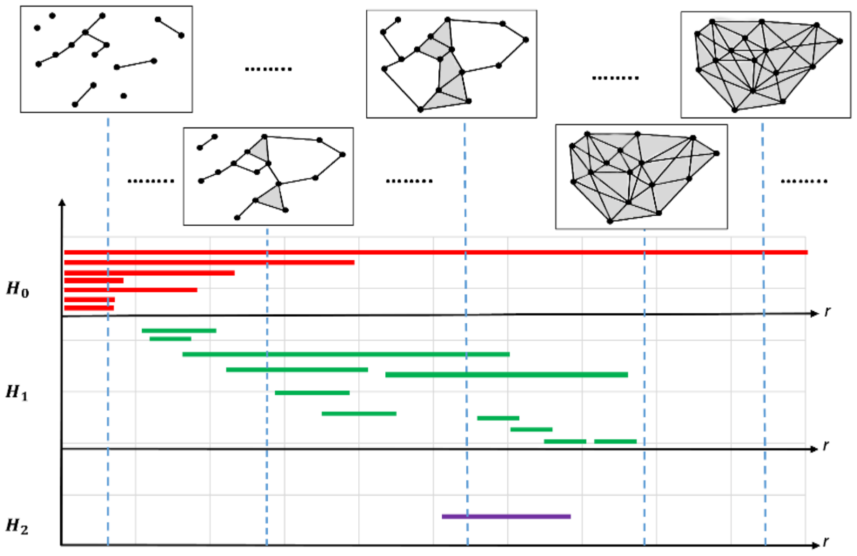

Figure 1- Persistence Barcode.

Back to the performing of persistent homology computation, there are several methods to build the filtered simplicial complex for the point cloud. For doing so, we utilized the Vietoris-Rips (VR) and the Alpha-Complexity algorithm (A_r)[35]. The following defines the VR [26].

Definition 1.3. VR is a method to create a filtered simplicial complex by utilizing equal distances between any two points. Let X be a dataset in the form of a point cloud and let $\epsilon > 0$, then define the VR complex as: $VR_\epsilon(X) = \{\sigma \subset X: \forall\, p, q \epsilon\, \sigma, d(p,q) \leq \epsilon\}$. If X has n points then the maximal possible number of VR complex is $(n-1)$- simplex which contains all points with all its sub-simplices in X. By taking set $\{\epsilon_i\}$ s.t $0 = \epsilon_0 < \epsilon_1 < \cdots < \epsilon_\iota$, then we have the filtered simplicial complex $X = VR_{\epsilon_0}(X) \subset VR_{\epsilon_1}(X) \subset \cdots \subset VR_{\epsilon_\iota}(X)$.

On the other hand, since A_r is the nerve of restricted Voronoi regions cover, and a subcomplex of Delaunay triangulation. Thus, To own a deep imagination and a better understanding of A_r, the approaches of Voronoi diagrams and Delaunay triangulation should be preferably clarified.

Definition 1.4. A Voronoi diagram, as shown in Figure 2 (A), partitions the space X with areas close to a set of finite points $S \subseteq X$, these points called sites u. Each site corresponds to an area called Voronoi cell v_u, and the Voronoi cell of u is defined as: $v_u = \{x \in X: d(x,u) < d(x,v), v \in S\}$ [36].

Definition 1.5. Delaunay triangulation, as shown in Figure 2 (B), state that between any two points $u, v \in S$ there is an edge if and only if their Voronoi cells V_u and V_v have a common edge in the Voronoi diagram [37].

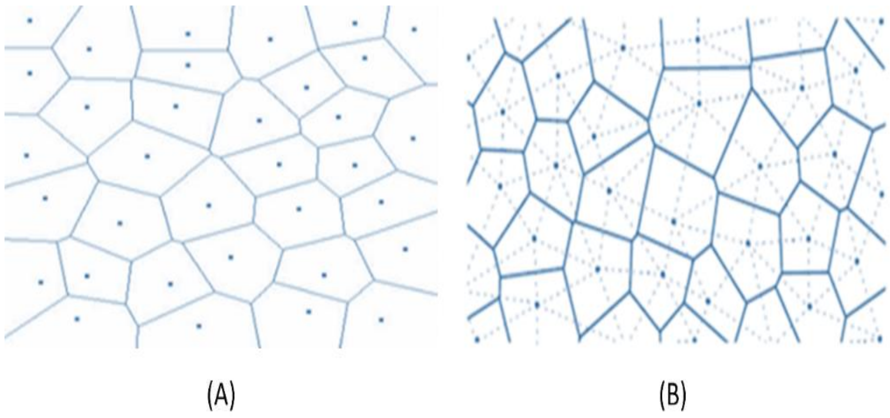

Figure 2- (A): Voronoi Diagram, (B): Delauany Trianulation.

Definition 1.6. Thes alpha complexes of a space X, denoted by $A_r(X)$, is a subcomplex of Delaunay triangulation. Let $S \subseteq X, u \in S$ and let $r > 0$, $X_r = \bigcup_{u \in S} B(u,r)$, where $B(u,r)$ be a closed ball with radius r and center u. Additionally, let V_u be a Voronoi cell, then alpha complexes are as follows: $A_r(X) = \{S \subset X: \forall u \in S, \bigcap_{u \in S}(V_u \cap B_u(r)) \neq \emptyset\}$. r with a very small value leads to the situation in which no collide occurs among Voronoi cells; consequently, the complex is the same as the cloud point. A_r is the Delaunay triangulation when a very large r is taken. If r is moderate in between, we get different complexes, i.e, $0 = r_0 < r_1 < \cdots < r_i$, then we have filtered simplicial complex $X = A_{r_0}(X) \subset A_{r_1}(X) \subset \cdots \subset A_{r_i}(X)$ [37].

By the end of this section, we have described one of the powerful methods in the art of computational topology, and with the aforementioned methods founded, we now have the tools to examine the expressivity of MLPR in the domain of topological data analysis.

4. Proposed Methodology

The initialization of an effective architecture design is the most significant part to improve ANN performance. In this section, we are about to explain the applying of persistent homology to characterize the efficiency of MLPR architectures. The TC of a training dataset impacts the expressivity of MLPR. More specifically, an MLPR with simple architecture would fail to express a dataset with high TC. In contrast, a dataset with simple TC does not require a more complicated architecture in order to be expressed. In domain of architecture selection, If a search space contains a group of architectures that some of might be suitable for expressing the training dataset's homology, elimination of the rest that incapable to express the dataset's persistent complexity diminishes the search space significantly, and thus help in reaching the optimal architecture. Thus, diminishing of a search space should be started by computing the dataset complexity. In the forthcoming paragraphs, we have shown how to install a simple effective approach to compute the persistent homology of a dataset, which passes through three phases; the reduction phase, the construction phase, and the extraction phase.

Consider that S is a high-dimensional training dataset, has a specific TC, which needs to be forwarded to MLPR. before selecting the architectural parameters (depth, width) of MLPR, dataset S is embedded with a useful extractable homology that may help in specifying a better capacity of MLPR. To extract such homology, the reduction phase is carried out, in which the high dimensions of S should be reduced to 3-dimensions by employing a dimensionality reduction technique.

The dimensionality reduction for high-dimensional space X in a low-dimensional space Y with a small error nearly preserves Persistent homology. As several data-embedding algorithms try to be homeomorphism between X and Y that is $X \cong Y$ by corollary 1.1 we have $H_n(X) \cong H_n(Y)$. Thus, thereby overcoming the impediments of existing persistent homology algorithms which do not operate well with high-dimensional datasets. A dimensionality reduction method t-SNE is used to achieve this matter.

After reducing the dimensions, the construction phase takes place, in this phase, the filtered simplicial complex of the reducted S is computed by utilizing a filtration method. Regarding this matter, the A_r algorithm, described in section 3, is exploited to construct the filtered simplicial complex of the dataset S, and thus, at the end of filtration, the topological features of S are stored in the form of a persistence barcode to be ready for computed the persistent homology by extracting the BN. Since dataset S is embedded in 3D form, The persistence barcode will curry the topological information of three features which are clusters, circles, and gaps.

In the extraction stage, the BN of S should be extracted from the barcode. There are a few extracting methods for doing such matter; However, In our work, we proposed a method that depends on the principle of *interval*. In our barcode, a fixed-length interval scans the entire barcode with a fixed radius r that can pack the prominent topological features, stores them in a BN vector. in more detail, we initially select a fixed r value which represents how many units would be inside the decision boundary of the scanning interval on the barcode's x-axis. Thus, starting from the left endpoint of the barcode, the BN of the topological features is computed at the initial interval boundary which is $[0, r]$. At every scanning step towards the right endpoint of the barcode, the interval boundary increasing by a specific real value, and the BN of the topological features is computed accumulatively to be stored in a vector $[\beta_0, \beta_1, \beta_2]$ in such a way that any previously arisen features will not be considered in the accumulation at the current interval. This process goes on till reaching the right endpoint of the barcode at which the final BN is obtained.

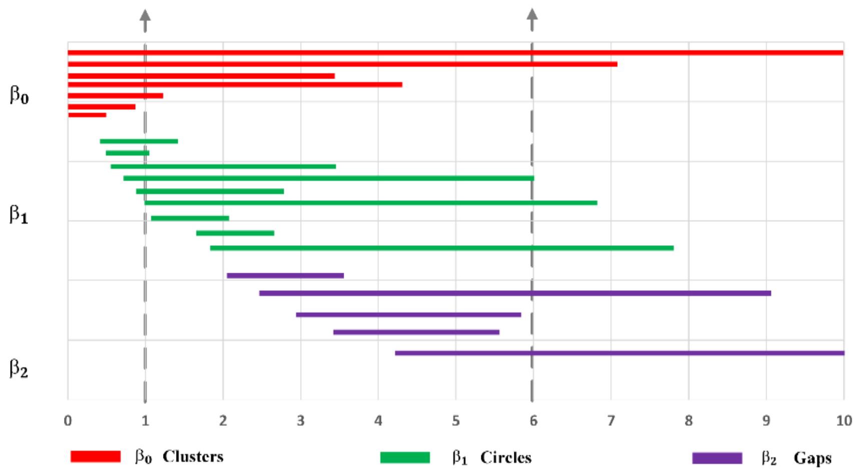

Figure 3-Simple Barcode Example.

To further clarify our method, Figure 3 shows a simple 3D barcode that is ready to extract BN out of it. Suppose we choose $r = 5$, then we compute the BN of the topological features at the first interval $[0,5]$, and store that BN in a vector $[\beta_0, \beta_1, \beta_2]$. Here, the BN represents every feature bar that passes through the entire current interval, which means it does not appear or die inside that interval. For $[0,5]$, BN contains only two clusters that pass through the current interval that makes the BN vector stores $[2,0,0]$. The scanning interval moves right by one unit to reach $[1,6]$ at which there are two cycles penetrate it, along with the previously appeared two clusters. Hence, the BN would be $[2,2,0]$. Table 1 has shown the BN extracting for the rest of the interval, besides, the final BN was also determined during the last interval, which was $[2,3,2]$. Note that the scanning step does not have to be an integer value, but it depends on the complexity of the barcode.

Table 1- BN for Each Interval

Interval	Betti Number
[0,5]	[2,0,0]
[1,6]	[2,2,0]
[2,7]	[2,3,0]
[3,8]	[2,3,1]
[4,9]	[2,3,1]
[5,10]	[2,3,2]

After discovering the complexity of the dataset, the next stage is to train the MLPR through one layer with an increasing width to detect the one layered architecture through which MLPR can express the TC of the dataset. Turning to the depth, MLPR has to be trained with search spaces N consisting of $n * m$ architectures in which n is depth size; m is width size. By using such search space we expect to be able to discover the relationship between data complexity and MLPR architectures.

5. Methods Setup and Results

In this section, we have described the applied datasets, computed results of persistent homology, and the preparation of MLPR. Furthermore, we have shown several MLPR training results that have been subjected to various datasets complexities. All algorithms were programmed utilizing PYTHON 3.8 in Jupyter within the ANACONDA environment[1].

5.1 Datasets description

In 2018, we got lucky that a huge data of Volve field was unveiled by oil company in Norway; thus it was a good opportunity for researchers, IT companies, and universities to utilize this data to solve several oil issues. The Volve Data Village data set used for this study is a licensed material and subjected to the Terms and Conditions. The intellectual property copyrights of the dataset are owned by Equinor, Exxon Mobil Exploration & Production Norway AS, and Bayerngas Norge AS. The author of this study is licensed to use the materials and affirm that there is no endorsement for this study made by Equinor and the former volve partners. The Volve field is located in the Norwegian North Sea, with a field life of 2008-2016, produced 10,037,081 SM3 of oil from Hugin Formation. 21 wellbores were drilled and logged with different tools to reveal some of the geological and petrophysical information. This information is important to decide upon the completion and the depletion plan of the oil field. The well logs of Volve Field were used in this study to understand the topology of the Volve filed data as a function of Neural Network topology.

On the other hand, since MLP cannot train with missing values in the datasets, thus, before starting the training process of MLP, it is viable to clean out our datasets that contain several empty records. Therefore, the dataset was examined and prepared to be suitable for our model. After the data cleaning phase, with an attempt to be varying in terms of shape and amount, we have extracted five random datasets samples $D_1, D_2, \dots, D_5$ from the full Volve filed dataset such that $\forall D_i \in \mathbb{R}^{m\times n}$ having n dimensions (features) and m observations (examples), where $i = 1, \dots ,5,$ and $n = 9$. The number of data points in each dataset is denoted by $N({D'}_i)$, see the forthcoming table. The dataset's nine features are Gama Ray, caliper, resistivity, neutron porosity, bulk density, density correction, measured depth, True vertical depth, and P-sonic.

On the other hand, we have considered the first 8 features as inputs for training every architecture of MLP regressor to predict a single output which is the P- sonic log. Furthermore, those features were reduced to 3 inputs using t-SNE method and supplied to the A_r algorithm, as an input vector. Thus, we denote those training datasets as ${D'}_1, {D'}_2, \dots, {D'}_5$, such that $\forall {D'}_i \in \mathbb{R}^{m\times 8}$, $i = 1, \dots ,5$.

5-2 Persistent Homology Setup and results

Before implementing any persistent-related step, all five datasets need to be normalized. For this purpose, we have utilized the Yeo Johnson Power Transformation method. As previously declared, the dimensions of datasets have to be reduced to 3D utilizing t-SNE. Figure 4 shows the dimensionality reduction of ${D'}_1, {D'}_2, \dots, {D'}_5$.

The datasets are now ready to be passed to the A_r algorithm for the filtration procedure. It is to mention that we initially wanted to utilize the VR algorithm but, unfortunately, our computer suddenly stopped working since VR required high computations because it performs the distance formula on every pair of data points in the dataset. In contrast, the A_r algorithm is characterized by dealing only with the Voronoi cells and closed balls. Thus, we have concluded that VR is suitable when datasets are small and trivial if high capabilities of cloud computing are not available. On the other hand, A_r is away better approach that can deal effectively with a huge dataset. After we had executed the A_r algorithm on D', the results of simplicial complexes $\alpha({D'}_i)$ for the five datasets were stored in the form of the persistence barcode as shown in Figure 5. Then, We have extracted the BN, $\beta({D'}_i)$, utilizing our barcode reading method. Table 2 shows all the results along with the final homologies.

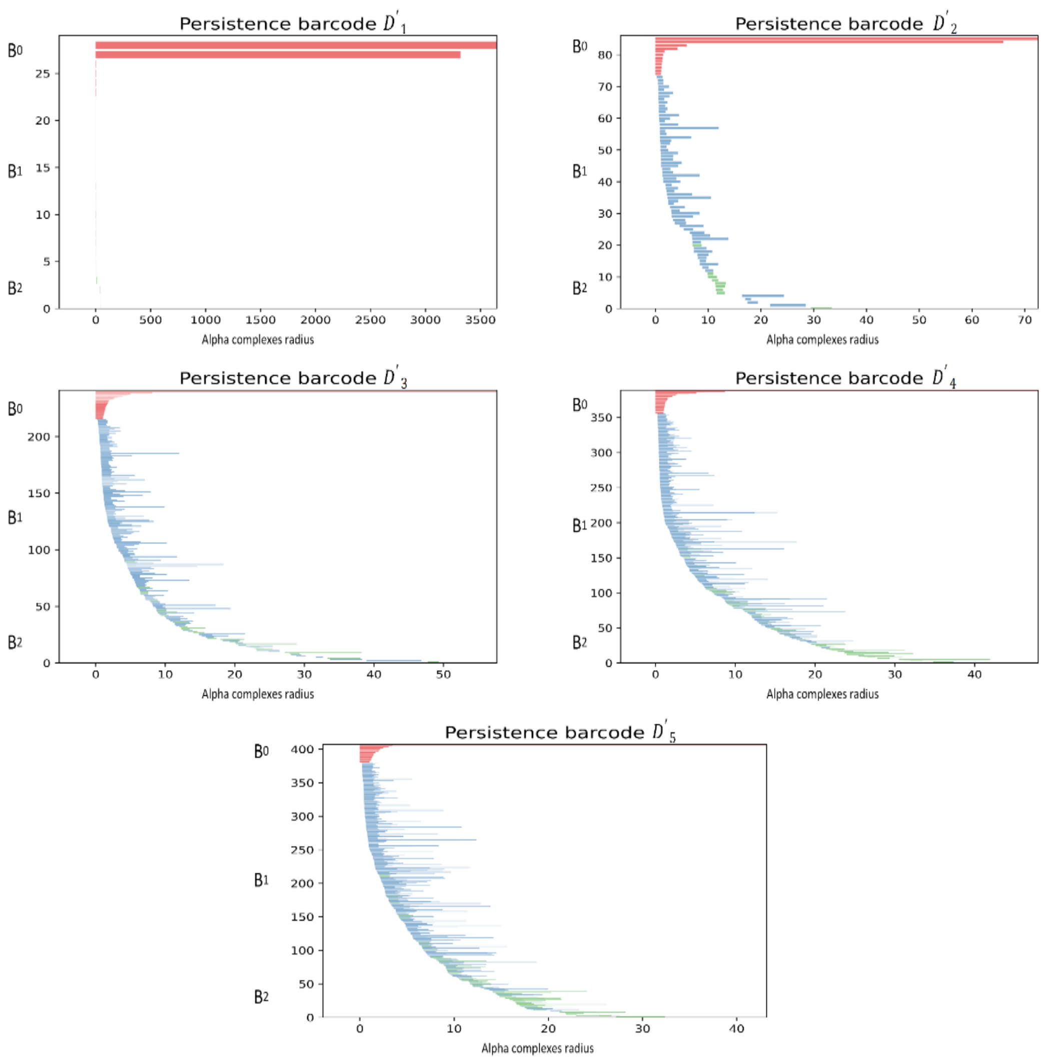

Figure 4- Persistence Barcode for Datasets.

Table 2- BN and Homology Groups

DataSets	$N(D'_i)$	$\alpha(D'_i)$	$\beta(D'_i)$	$H(D'_i)$
D'_1	1000	27461	[2,0,0]	$[\mathbb{Z}^2, \mathbb{Z}^0, \mathbb{Z}^0]$
D'_2	3000	83163	[2,1,0]	$[\mathbb{Z}^2, \mathbb{Z}^1, \mathbb{Z}^0]$
D'_3	7000	199089	[1,4,1]	$[\mathbb{Z}^1, \mathbb{Z}^4, \mathbb{Z}^1]$
D'_4	15000	425033	[1,7,1]	$[\mathbb{Z}^1, \mathbb{Z}^7, \mathbb{Z}^1]$
D'_5	20000	560473	[1,5,1]	$[\mathbb{Z}^1, \mathbb{Z}^5, \mathbb{Z}^1]$

5.3 MLPR setup and results

To examine the influence of data homology on the ability of MLPR architectures expressivity, we evaluated MLPR architectures by their ability to learn and express the homology on datasets of different TC with a growing number of layers and hidden units.

In the beginning, we have shown the difficulty level of MLPR width to express the homology of our five datasets by exploring the training error of MLPR architectures, each with a single layer and an increasing width $h = \{1,2,\dots,60\}$, where h is the number of neurons. Furthermore, we have analyzed the ability of MLP architectures to express the homology of our datasets by considering an increasing depth $l = \{1,2,\dots,7\}$ and width

$h_i = \{1,2,\ldots,60\}$, where $1 \leq i \leq l$. Hence, a fully connected architecture with RELU activation function has been considered to implement such analyses. For each architecture, The input layer was fed with the 8 well logging features and expected to predict one feature which is the P-sonic log. For a suitable convergence, We have run Adam optimizer for 200 iterations with a momentum of 0.9 and a fixed learning step of 0.01 to minimize the objective.

For understanding all the blow figures, we have considered the training error of 0.044 as an expectable error for determining whether an architecture has succeeded to express the homology. Hence, if the training errors of many consecutive architectures were stable under 0.044, we can say that those architectures own the best architectural capacity to express the dataset concerned.

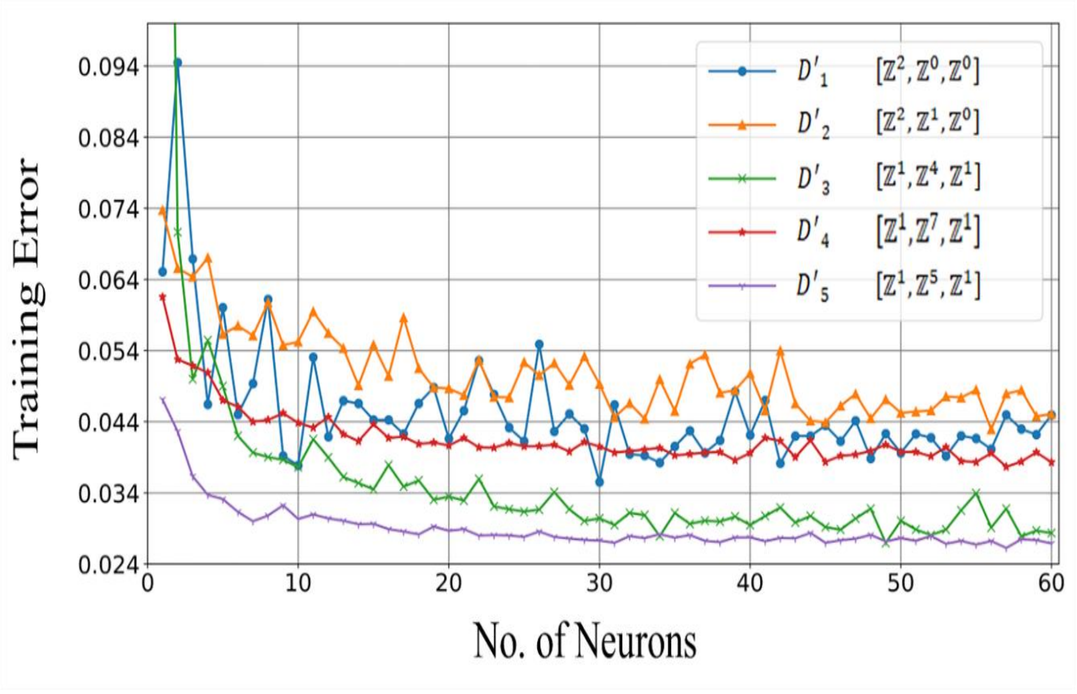

Figure 5-Training Error VS MLPR Architectures with Increasing Neurons.

Figure 5 shows the final training errors (loss function values) for each MLPR architecture with a single layer and increasing neurons till 60. It is obvious at first sight of training error that a dataset with less complexity has a better chance to capture the best architecture so early. Since D'_2 was the more complex dataset, architectures have struggled through the entirety of the above figure to reach the training threshold. As a result, it requires more than 60 hidden neurons for D'_2 to stable under the training threshold. Throughout the above figure, there is a jostle between D'_4 and D'_1. However, D'_4 has reached the training threshold at architecture with 13 neurons and continuously owned stability under that threshold till the end due to simplicity in its homology compared to D'_1. In contrast, since D'_1 is embedded with two clusters with a very long lifespan, as shown in its barcode, it was expressed by the architectures with 42 neurons to 60. On the other hand, the existence of seven circles and one gap on D'_4 has made it more complex than D'_5 and D'_3. Finally, D'_5 and D'_3 homologies have been successfully expressed with a less number of neurons since they have the most uncomplex homology in the set. A fewer number of neurons was able to express the homology of D'_5 compared to D'_3. However, both of those datasets have nearly the same complexity with a slight simplicity to the D'_3 as we see in the next figure.

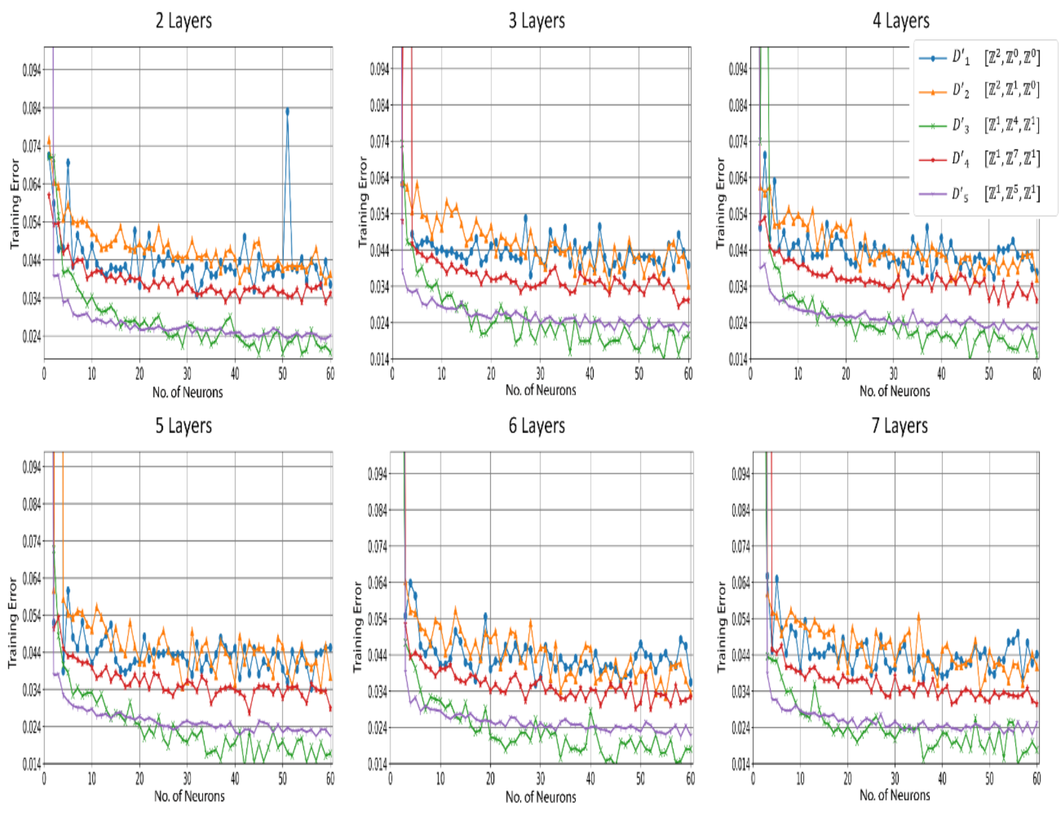

Figure 6-Training Error for Each MLP Architectures with Increasing Neurons and layers.

Figure 6 Shows the training errors of increasing neurons and layers. Our goal is to detect the best number of layers that make the architecture stable under the training threshold. As the layers increases, a huge number of architectures have gained a training error under the threshold, which is 0.044 for all datasets. Furthermore, the more complex the homology, the more layers it requires to reach a low training error. For D'_1, the majority of architectures with two and three layers were stable under the threshold. Thus, in order to acquire a training error below the threshold error, D'_1 can be content with two or three layers, and there would be no need to increase the computational complexity with more layers unless a lower acceptable error is targeted such as 0.034.

In the same vein, since D'_2 has the more complex homology, architectures with four layers have perfectly expressed the homology of D'_2, it seems fairly obvious in the architectures in between 30 to 60 neurons. On the other hand, D'_4, D'_3, and D'_5 , which are the simplest datasets, have possessed stability under the acceptable error early in all layers; thus, no high number of layers is required to express their homology. Nevertheless, if we change the acceptable error to a lower value such as 0.024, we can see that D'_4 and D'_5 require more layers to stable under such error, but not D'_3, which has owned the stability under this low acceptable error even in four layers due to the simplicity in its homology which is the simplest homology among all datasets.

From the above figures, we also have concluded that if a dataset has more than one cluster on its homology, a few architectures would possess stability which can be noticed through the zigzag of $D'_1$and D'_2. Thus, clusters significantly affect the training process of regression hyperplane, force the training errors of architectures to behave like a zigzag. hence, the number of consecutive architectures that successfully express the homology would be very small. It is obvious that increasing the number of layers reduces the training error; however, to prevent selecting the architectures that exist at the peak of zigzags, the more clusters, the more careful should consider during selecting the architecture. In contrast, architectures that attempt to express a homology with only circles or gaps dose not suffer any zigzag. Consequently, increasing the number of layers, commensurate with the number of circles and gaps, is an outstanding idea to get an acceptable training error.

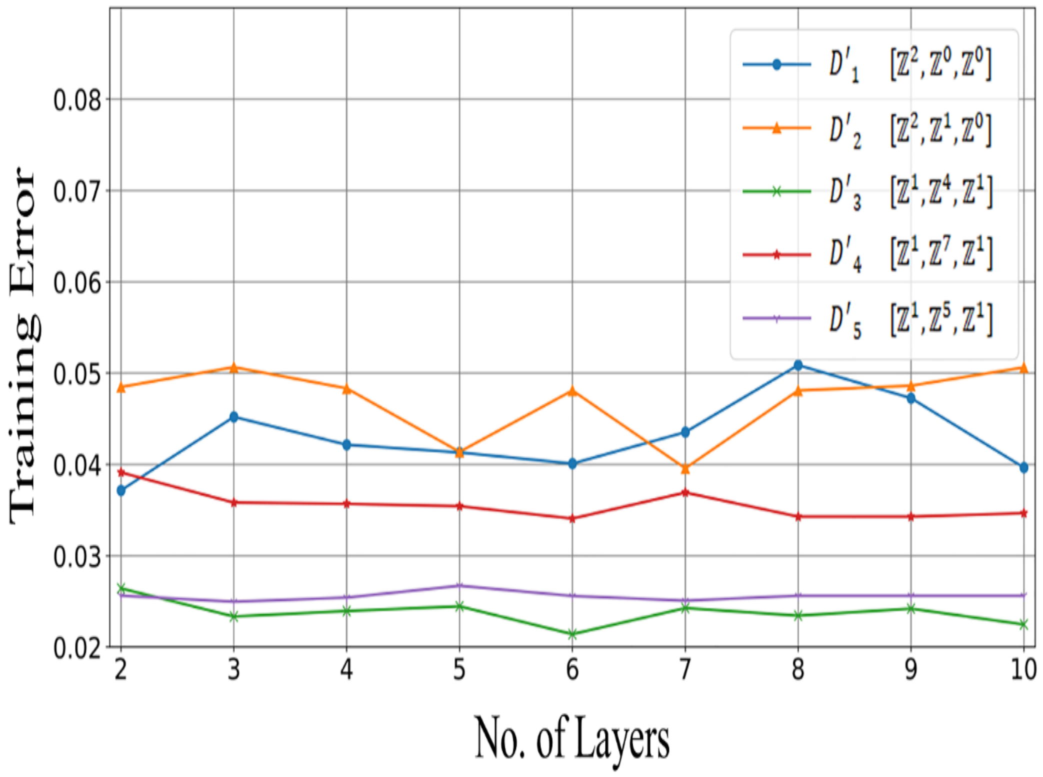

Figure 7-Training Error VS MLPR Architectures with Increasing Layers.

In Figure 7, we have set the number of neurons to 20, increased the layers to 10 in order to further clarify the relationship of changing layers with the ability of architectures to express homology. In short, the dataset with high complexity would struggle to find a better training error, especially, when embedded with more clusters. Furthermore, the more complex the dataset, the more layer it requires to reach a better training error than other datasets. Based on Figure 7, Table 3 shows the depth of each architecture at which the homology of each dataset was expressed successfully with the lowest training error.

Table 3- Architecture Expressivity.

DataSet	$H(D'_i)$	Training Error	Layer
D'_1	$[\mathbb{Z}^2,\mathbb{Z}^0,\mathbb{Z}^0]$	0.0373	2
D'_2	$[\mathbb{Z}^2,\mathbb{Z}^1,\mathbb{Z}^0]$	0.0393	7
D'_3	$[\mathbb{Z}^1,\mathbb{Z}^4,\mathbb{Z}^1]$	0.0216	6
D'_4	$[\mathbb{Z}^1,\mathbb{Z}^7,\mathbb{Z}^1]$	0.0340	6
D'_5	$[\mathbb{Z}^1,\mathbb{Z}^5,\mathbb{Z}^1]$	0.0249	3

As a conclusion from all the above empirical analysis, when training a dataset, instead of randomly initializing the architectural parameters of MLPR, it is better to have a look at the dataset's homology to extract useful information that leads to diminishing the search space of architectures. under this theory, we have concluded that a dataset with more clusters in its homology has to be expressed with an architecture that provides a severe nonlinearity by extending the depth and width of MLPR to reach an acceptable training error. This is due to the suffering of the MLPR hyperplane to fit on the clusters data points. In contrast, circles in data homology have a lower impact on MLPR training ability than clusters. Thus, circles can be treated with a moderate consideration in determining the architecture, thereby fixing the size of depth, and increasing the width units might be a robust behavior to reach an acceptable training error without exposing MLPR to high computations. On the other hand, gaps make no high distinctness on the MLPR training process, and thus, concentrating on the impacts that clusters and circles make are more essential than wasting resources and time on low impact features.

It is worthy to draw the attention of readers that the abovementioned methodology can be considered as progress toward finding a notion that takes the topological features of the training dataset to outputs the optimal, or nearly optimal, depth and width of an MLPR. For finding such a notion, a huge amount of various datasets along with cloud computing capabilities must be available for further analysis.

6. Conclusion

In this paper, we considered the problem of neural networks architecture's initial selection. This problem is very popular in deep learning community. Usually, in deep neural networks, hidden neurons and layers are selected randomly, or with redundance numbers of neurons and layers. Several mathematical tools have been proposed to avoid the imperfections of those two categories of selections. The majority of those tools have adopted statistical methods to define a relation between dataset and architecture, others adopted a gradient-based, evolution, and swarm optimization algorithm to detect the optimal architecture. We have considered this problem from an algebraic topology perspective by utilizing the topological complexity of the dataset that is, the optimality of architecture can be measured by its ability to express the homology of that dataset. For measuring the complexity of datasets, we have dive into the topological data analysis field which provides robust tools to measure the data complexity, and thus, we have proposed a compound algorithm to compute the persistent homology that takes advantage of three methods which were t-SNE method, alpha complexity method, and a persistence barcode reading method for extracting the Betti number. We applied this method to MLPR with different data complexities and various architectures. Our empirical analysis has shown that MLPR suffers to express the high complex dataset, it requires more (neurons/layers) to reach low training error. In contrast, the less data complexity, the fewer (neurons/layers) it requires to reach a better capacity. Thus, using such a methodology will avoid the arbitrary selection of ANN's architecture and diminishing architectures of the search space. Our methodology can be exploited with the help of cloud computing to find an equation that may be utilized to set an optimal initial architecture as we will attempt to do in the future.

References

[1] A. Bashar, "Survey on evolving deep learning neural network architectures," *Journal of Artificial Intelligence*, vol. 1, no. 02, pp. 73-82, 2019.

[2] H. Greenspan, B. Van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1153-1159, 2016.

[3] H. M. Ahmed and H. H. Mahmoud, "Effect of Successive Convolution Layers to Detect Gender," *Iraqi Journal of Science*, vol. 59, no. 3C, pp. 1717-1732, 2018.

[4] Z. A. Khalaf, S. S. Hammadi, A. K. Mousa, H. M. Ali, H. R. Alnajar, and R. H. Mohsin, "Coronavirus Disease (COVID-19) Detection using Deep Features Learning," 2020.

[5] G. S. Babu, P. Zhao, and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *International conference on database systems for advanced applications*, 2016: Springer, pp. 214-228.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[7] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning* (no. 2). MIT press Cambridge, 2016.

[8] S. K. Abaas and L. E. George, "The Performance Differences between Using Recurrent Neural Networks and Feedforward Neural Network in Sentiment Analysis Problem," *Iraqi Journal of Science,* pp. 1512-1524, 2020.

[9] M. Kubat, *An introduction to machine learning*. Springer, 2017.

[10] I. S. Alshawi, M. H. K. Jabbar, and R. Z. Khan, "Development of Multiple Neuro-Fuzzy System Using Back-propagation Algorithm," *International Journal of Management & Information Technology,* vol. 6, pp. 794-804.

[11] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1-9.

[12] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research,* vol. 9, no. 11, 2008.

[13] S. Ayesha, M. K. Hanif, and R. Talib, "Overview and comparative study of dimensionality reduction techniques for high dimensional data," *Information Fusion,* vol. 59, pp. 44-58, 2020.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.

[15] M. G. Augasta and T. Kathirvalavakumar, "A novel pruning algorithm for optimizing feedforward neural network of classification problems," *Neural processing letters,* vol. 34, no. 3, p. 241, 2011.

[16] D. Chhachhiya, A. Sharma, and M. Gupta, "Designing optimal architecture of recurrent neural network (LSTM) with particle swarm optimization technique specifically for educational dataset," *International Journal of Information Technology,* vol. 11, no. 1, pp. 159-163, 2019/03/01 2019, doi: 10.1007/s41870-017-0078-8.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems,* vol. 25, pp. 1097-1105, 2012.

[18] S. Saxena and J. Verbeek, "Convolutional neural fabrics," *arXiv preprint arXiv:1606.02492,* 2016.

[19] R. Pascanu, G. Montufar, and Y. Bengio, "On the number of response regions of deep feed forward networks with piece-wise linear activations," *arXiv preprint arXiv:1312.6098*, 2013.

[20] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE transactions on neural networks and learning systems,* vol. 25, no. 8, pp. 1553-1565, 2014.

[21] W. H. Guss, "Deep function machines: Generalized neural networks for topological layer expression," *arXiv preprint arXiv:1612.04799,* 2016.

[22] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," *arXiv preprint arXiv:1606.05340*, 2016.

[23] W. H. Guss and R. Salakhutdinov, "On characterizing the capacity of neural networks using algebraic topology," *arXiv preprint arXiv:1802.04443,* 2018.

[24] R. Rabadán and A. J. Blumberg, *Topological data analysis for genomics and evolution: topology in biology*. Cambridge University Press, 2019.

[25] A. Hatcher, *Algebraic topology*. 清华大学出版社有限公司, 2005.

[26] M. Feng and M. A. Porter, "Persistent homology of geospatial data: A case study with voting," *SIAM Review,* vol. 63, no. 1, pp. 67-99, 2021.

[27] J.-D. Boissonnat, F. Chazal, and M. Yvinec, *Geometric and topological inference*. Cambridge University Press, 2018.

[28] S. MacLane, *Homology*. Springer Science & Business Media, 2012.

[29] H. Adams *et al.*, "A fractal dimension for measures via persistent homology," in *Topological Data Analysis*: Springer, 2020, pp. 1-31.

[30] J. M. Curry, "Topological data analysis and cosheaves," *Japan Journal of Industrial and Applied Mathematics,* vol. 32, no. 2, pp. 333-371, 2015.

[31] G. Carlsson, "Topology and data," *Bulletin of the American Mathematical Society,* vol. 46, no. 2, pp. 255-308, 2009.
[32] M. Hajij, G. Zamzmi, and F. Batayneh, "TDA-Net: Fusion of Persistent Homology and Deep Learning Features for COVID-19 Detection in Chest X-Ray Images," *arXiv preprint arXiv:2101.08398,* 2021.
[33] R. J. Adler, O. Bobrowski, M. S. Borman, E. Subag, and S. Weinberger, "Persistent homology for random fields and complexes," in *Borrowing strength: theory powering applications–a Festschrift for Lawrence D. Brown*: Institute of Mathematical Statistics, 2010, pp. 124-143.
[34] M. N. Atienza Martínez, R. González Díaz, and M. Rucco, "Separating Topological Noise from Features Using Persistent Entropy," in *STAF 2016: Collocated Workshops: DataMod, GCM, HOFM, MELO, SEMS, VeryComp (2016), p 3-12*, 2016: Springer.
[35] C. S. Pun, K. Xia, and S. X. Lee, "Persistent-Homology-based Machine Learning and its Applications--A Survey," *arXiv preprint arXiv:1811.00252,* 2018.
[36] M. Sainlot, V. Nivoliers, and D. Attali, "Restricting Voronoi diagrams to meshes using corner validation," in *Computer Graphics Forum*, 2017, vol. 36, no. 5: Wiley Online Library, pp. 81-91.
[37] H. Edelsbrunner and J. Harer, *Computational topology: an introduction*. American Mathematical Soc., 2010.