Received 26 April 2016; | Revised 6 June 2016; | Accepted 6 June 2016

DOI: xxx/xxxx

RESEARCH ARTICLE

Utilization of the fractal dimension metric in training of dense Neural Networks

Kalin Stoyanov*[1] | Jordan Hristov[2]

[1]Department of Automation, University of Chemical Technology and Metallurgy, Sofia, Bulgaria

[2]Department of Chemical Engineering, University of Chemical Technology and Metallurgy, Sofia, Bulgaria

Correspondence

*Kalin Stoyanov, Department of Automation, University of Chemical Technology and Metallurgy, 8 Blvd. Sv. Kl. Ohridksi, Sofia 1756, Bulgaria. Email: kalin.stoyanov@uctm.edu

Summary

The process of training of Artificial Neural Networks essentially is optimization of the values of the weights w_{pq} associated with the arcs, connecting the nodes of the layers. This is a process of minimization of the Loss function (maximization of Accuracy function). During the training, the training data set recursively is utilized at subsequent stages, called *Epochs*. The training continues until a satisfactory values of the Loss, Accuracy etc. parameters are reached. The matrices W^{UV} comprising the weights of the arcs connecting the layers U and V, can be regarded as gray-scale images of a surface. Starting as random matrices, processed by recursive procedures, they gradually become fractal structures, characterized with respective fractal dimension D_f. In the presented article we have made an attempt to utilize the correspondence of D_f with the Loss/Accuracy values, in order to forecast the optimal ending point of the NN training process. Similar conclusions were made for the correspondence between the number of layer's nodes and D_f.

An attempt to apply statistically more rigorous approach in the determination of the slope of the regression line in Richardson-Mandelbrot plot, was made.

KEYWORDS:

Fractal dimension. Neural networks, Richardson-Mandelbrot plot

1 | INTRODUCTION

Along with the quantum mechanics, the general theory of relativity and the double helix model of DNA, the fractal analysis has been suggested as one of the four most significant concepts of 20^{th} century[1]. The fractal dimension[2] has found quite wide application as a metric for analysis of complex surfaces and remotely sensed images[3],[4],[5]. In the same time Deep Learning receives fast growing popularity as classification and regression modeling tool with increasingly importance both in science and in our everyday lives. Important areas as cancer diagnosis, precision medicine, self-driving cars, predictive forecasting, and speech recognition, were already significantly affected[6],[7].

The basic Artificial neural network (NN), represented as Multilayer perceptron (MLP), which is the backbone of the Deep Learning, can be regarded as a learning machine. NN is constructed of four main components - Layers (made of nodes), Weight matrices, activation function and the method to evaluate the performance of the neural network. The particular construction of these parts and their interrelations characterize the purpose and application of the MLP. While the most of the parts do not change during the process of training of the NN, the weight matrices gradually improve their elements, towards increase of the performance of the NN. The weight matrices, considered from an abstract point of view, can be regarded as $n \times n$ digital images.

[0]**Abbreviations:** MLP, Multilayer perceptron; NN, Neural Network, R-M plot, Richardson-Mandelbrot plot; IRF, RM line, Richardson-Mandelbrot regression line

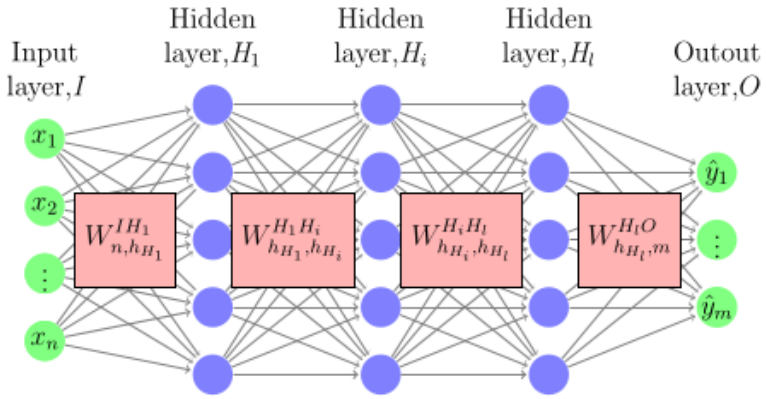


FIGURE 1 Fully connected Multi Layer Perceptron (MLP). The nodes of the terminal layers (I and O) are green. The nodes of the Hidden Input and Hidden Output layers (H_1 and H_4) are red, while the Hidden layers 2 and 3 are blue.

This similarity between the weight matrices of NN and digital images, would allow us to employ the extensive fractal analysis toolbox, developed for handling ans analyzing such objects.

In this paper we have made an attempt to exploit the analogy of the NN weight matrices with the fractal objects, so we can employ the methods for evaluation of the fractal dimension. A tool for suggesting the length of training of the NN, namely the number of *Epochs* was proposed.

In the course of the work, we have noticed that some authors expressed their concerns with the reliability of the Richardson-Mandelbrot[2,8] (RM line). Clarke[9] express their concerns regarding the quality of the regression equation, which concerns of course the regression coefficient b. Also in[10] comments the instability of the regression line, which can be biased by the disproportional big number of points at its end. As long the RM line and particularly its slope, plays a vital role in the presented work, we proposed some statistical more rigorous approach in calculating the slope of the line, which is closely related with the fractal dimension $D_f = 1 - b$,[11].

According to the online edition of Collins English Dictionary[12], *Fractality* is *the quality of being fractal or subdivided*. In the present article we will show the usage of a method to evaluate the fractality of an object particularly a weight matrix of a Multi-Layer Perceptron. Also we will utilize it as a tool to control resources, needed for the training stage of a Multi-Level Perceptron.

We claim that in the course of the training of the NN, the weight matrices reach a state of fractality. To monitor the process of achieving the fractality state of the weight matrices we use the behavior of the slope of the RM line. We noted that is is pointless to continue the training the Neural network, after the stage where the regression coefficient of the RM line cease its initial severe fluctuations. Also it was observed that (at least considering the working example of MLP with two hidden layers) the increase of the nodes (another resource consuming factor), does not improve the NN performance in terms of Loss and Accuracy.

In the following Section 2, we will provide a brief description of the structure of the Multi Layer Peceptron (MLP). Then in Section 3 we will present the Fractals dimension as a tool to characterize complex objects. In Section 4 we will discuss the fractality of the NN of MLP type and how to utilize it to control the resources spent for training of the NN.

2 | NEURAL NETWORKS

In the present article we will consider artificial Neural Network (NN)[13], as a tool for solving of classification tasks. We believe that the conclusions presented here can be extended also to the regression tasks domain as well. Here, we will discuss fully connected deep neural networks[13], which also will be addressed as multi-layer perceptrons (MLP). As a training protocol we will apply the backpropagation with gradient descent algorithm.

NN are constructed of three types of layers - input layer, designated as I, hidden layers, $H_i, i = 1, l$ and an output layer O, see Figure 1 , where a typical MLP is illustrated. Each layer is constructed of arbitrary number of nodes. Each node of layer

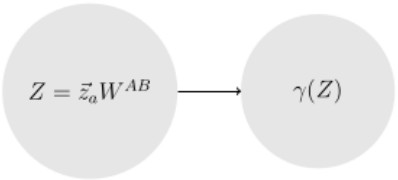


FIGURE 2 Computational graph, representing the calculations performed within each node of the neural network, representing the computational graph of each node of some computational layer. The vector $\vec{z}_A$ representing the output data from the a_{th} layer, is multiplied by the weight matrix W^{AB} associated with the arcs, connecting the layers A and B.

$j-1$ is connected with a directed arc to a node belonging to the layer j. The layer next to I is the first hidden layer, while the the hidden layer preceding the output layer O, is the last one.

Apart of the layers I and O, which represent the input and output data matrices X and $\hat{Y}$, the rest of the layers are paired with weight matrices. The hidden layers also will be addressed here as computational ones.

Each node from the computational layers, can be represented as a directed computational graph of two nodes $\vec{z}_a W^{AB}$, Fig (2). In the MLP context, usually the function $\gamma(\cdot)$ represents the conventionally used activation function.

Each layer is paired with a weight matrix, which precedes it in the NN construction. Only the input layer I is not paired. The dimensions of the weight matrices depends on the number of nodes in the layers, connected by the respective arcs. For instance (see Figure 1), the dimension of matrix $W_{4,5}^{IH_1}$ is $[4\times 5]$, of matrix $W_{3,5}^{H_1,H2}$ - $[3\times 5]$ and so on. The dimensions of the matrices W^{IH_1} and $W^{H_3H_4}$ are determined by the number of the features (number of the columns of X) and the number of classes to be classified among (number of the columns of $\hat{Y}$).

The matrix $X_{N\times n}$ contains the set of N patterns, subject to classification, described by n features. The matrix $Y_{N\times m}$ represents the set of labels associated with each of the patterns $x_i, i=1,N$. The row $y_i, i=1,N$ of Y contains a string of m values ε1ε and ε0ε, where, according to the utilized here *One Hot Encoding* [13] approach, by ε1ε is designated the class associated with the respective pattern, with ε0ε are designated the rest of the m classes. The rows $\hat{y}_i, i=1,N$ of $\hat{Y}$ contains the evaluation of the probability distribution, associated with the affiliation of the sample x_i to particular class. The values $\hat{y}_{i_j}, j=1,m$ designates the predicted values of probability, the sample x_i to belongs to the class $j, j=1,m$.

The classification tasks, considered here are solved by minimizing the *Loss* function

$$\Lambda \equiv \Lambda(Y,\hat{Y}), \tag{1}$$

where $\hat{Y}=\gamma(X,\mathcal{W})$, while $\mathcal{W}=\left\{W_{h_A h_B}^{AB}\right\}$ is a set of l weight matrices W, $\gamma(\cdot)$ is an arbitrary activation function.

The Loss function Λ measures the discrepancy between the "correct" labels Y associated with the samples X and the respective "predicted" labels $\hat{Y}$. The learning process consists of iterative adjusting the elements of the matrices in $\mathcal{W}$ in order to find such set $\mathcal{W}^*$, which minimizes $\Lambda(Y,\hat{Y})$. The iterations aiming to deliver $\mathcal{W}^*$ are called *Epochs*.

The *Loss metric* is calculated by using the Categorical Cross-entropy function [14] $C=1/N\sum_{j=1}^{N}\left[-\sum_j y_{i_j}\log(\hat{y}_{i_j})\right]$.

The *Accuracy metrics* evaluates in what fraction of samples the correct class, corresponds to the maximum probability in the respective row of $\hat{Y}$. Which is the fraction of the samples where the Kronecker delta δ_{ij} equals 1, where j is the correct class for a particular sample x_k and i is the index of the maximum value in $\hat{y}_k$.

The function $\hat{Y}=\gamma(\cdot)$, can be regarded as composite function [15], Appendix B:

$$\begin{aligned}\hat{Y} =&\gamma\left(X,W_{h_l h_{l-1}}^{l,l-1}\right)\circ\ldots\circ\gamma\left(X,W_{h_2 h_1}^{2,1}\right)\equiv\\ &\gamma\left(\ldots\left(\gamma\left(\gamma\left(\gamma\left(X,W_{h_2 h_1}^{2,1}\right),W_{h_3 h_2}^{3,2}\right),W_{h_4 h_3}^{4,3}\right)\ldots\right),W_{h_l h_{l-1}}^{l,l-1}\right).\end{aligned} \tag{2}$$

or

$$\hat{Y}=\gamma^{(l)}\left(X,W_{h_l h_{l-1}}^{l,l-1}\right) \tag{3}$$

where (l) denotes the number of hidden layers of the Neural Network, or equivalently the $l^l h$ iteration, if we consider (2) and (2) as iterative computational procedure.

One can simplify the notation in (3) to

$$\hat{Y}\equiv F^{(l)}\left(X,W^l\right)=F^{(l-1)}\left(X,W^{(l-1)}\right) \tag{4}$$

By examining (4), which represents the feed-forward pass, one can observe that it can be regarded as a dynamical system with X and W^0 as initial state.

Considering (2)-(4) the equation (1) can be expressed as

$$\Lambda = \Lambda\left(Y, \hat{Y}\left(\gamma\left(X, \mathcal{W}\right)\right)\right) \tag{5}$$

where $\lambda(\cdot)$ is the Loss function, $\gamma(\cdot)$ is the activation function, common for all hidden layers.

During the backward pass (addressed also as back-propagation) one recalculates the weight matrices, calculated during the forward pass,

After calculating the derivative of the activation function, one uses it to multiply it by the derivative of the activation function of the previous layer, then the result to multiply by the derivative of the layer before it, and so on, through all of the hidden layers.

$$W_1^{i+1} = \frac{\partial \gamma}{\partial W_2}\left(\cdots\left(\frac{\partial \gamma}{\partial W_{l-1}}\left(\frac{\partial \gamma}{\partial W_l^i}W_{l-1}^i\right)W_{l-2}^i\right)\cdots\right)W_1^i, i = 1, e, \tag{6}$$

where e is the number of the *Epoch* to be performed.

The goal is to adjust the weight matrix W_1, which will serve as initial state for the next *Epoch*.

One can observe that the equation (6) also can be regarded as a dynamical system with W_1 as initial state. Also by examining (6) and (4) one can notes that a second dynamical system (6) is embedded into (4).

The training process goes trough the following main steps, for more details see [13], [16]:

1. Evenly distributed random values of the elements of the matrix in W_1, associated with the first layer are generated;
2. One performs an *Epoch*.
   a. By using the set of samples X one performs forward and backward passes trough the network layers;
      i. During the forward pass by starting with W_1, one iterates trough the layers, gradually calculating layer-wise, the weight matrices $W_i, i = 2, l$, where $W_{i+1} = \gamma(X, W_i)$. Eventually one calculates the values of $\hat{Y}$ and respectively the value of $\Lambda(Y, \hat{Y})$;
      ii. During the backward pass one adjusts the values of the elements of the weight matrices $W_i, i = l, 1$, by using some optimization protocol (e.g. the *gradient descent* method). It starts with the values of W_l and continues back to the first layer's weight matrix - W_1;
   b. One evaluates the performance of the Neural network by using the Loss and Accuracy metrics, gained during the *Epoch*.
3. The algorithm continues until the desired number of *Epochs* is reached;

During each *Epoch*, based on the current initial matrix W_1, which in turn is a result from the previous *Epoch*, the whole set $\mathcal{W}$ of the weight matrices is recalculated, namely

$$\mathcal{W}_{i+1} = E(\mathcal{W}_i), i = 1, e \tag{7}$$

where by E one designated the operations performed during each *Epoch*.

3 | FRACTALS

The fractals [11] have several major characteristics, which distinguish them from the Euclidean objects - fractals possess a self-similarity between the object's parts and its overall shape; they do not have specific size; one can find infinite details at every point; the fractals can be created by following of a construction procedure or recursive algorithm [17].

The self-similarity, which is one of the most visually recognizable features of the fractals implies that a part of the fractal, when magnified to the size of the whole is visually similar the entire object. They are scale independent - when the scale is varied, the dimension remains unchanged.

One distinguishes between artificial (mathematical [18]) and natural fractals. The mentioned above characteristics differs in the natural fractals (Example for natural fractal is the length of the coast of Great Britain, while example for artificial fractal is the Koch curve [11]). As long the natural fractals are not strictly self-similar, they are called *statisitcally self-similar* [2], [17].

We know e.g. [19] that a fractal object can be generated by taking of an arbitrary object (not necessary a fractal) and by repeatedly applying a *transformation* function to the points within a region of space. If $P_0 = (x_0, y_0, z_0)$ is an initial point, each iteration of the transformation function F generates successive levels of detail with the calculations progress

$$P_1 = F(P_0), P_2 = F(P_1), P_3 = F(P_2), \ldots \tag{8}$$

We can use either deterministic or random procedures at each iteration. The transformation function may be defined in terms of geometric transformations or it can be set up with non-linear coordinate transformations and decision parameters.

Another important feature of the fractals is that [11], a fractal is a set, for which the Hausdorff-Besicovitch dimension D_f strictly exceeds the Euclidean topological dimension D_E.

For the purposes of this article, we will employ the following fractal definition. Some objects will be regarded as fractals if they posses the following major features:

- the value of the fractal dimension (Hausdorff-Besicovitch dimension) D_f exceeds the respective Euclidean (topological) dimension D_E, namely
$$D_f > D_E \tag{9}$$
- they are created by iterative procedure, which eventually provides the property of (statistical) self-similarity;

3.1 | Fractal dimension

Klonowski [18], distinguishes two types of fractals - mathematical and natural. For the purpose of this article we will use interchangeably the terms mathematical and artificial fractals, when it comes to fractals generated on the basis of mathematical formula or procedure.

3.1.1 | Fractal dimension of artificial objects

Mandelbrot [2], [11] postulates that the general formula for dimension of a lines is

$$L(\epsilon) = C\epsilon^{(1-D_f)}, \tag{10}$$

where ϵ is the size of the stick (or width of divider, used as measuring tool), D_f is the fractal dimension. C is a constant, which depends on the type of the line, while $L(\epsilon)$ is the measured line length, equal to the number of steps, performed with the stick.

Mandelbrot postulates also the following formula for estimation of the fractal dimension of a surface

$$S(r) = Cr^{(2-D_f)} \tag{11}$$

Here by r, one designates the measure unit for calculation of the surface, respectively $S(r)$ is the surface area.

Considering that $r = 1/d\sqrt{L}$ or $L = \frac{1}{r^{D_f}}$, one can calculates the dimension D_f of self-similar object by using the formula

$$D_f = 1 - \frac{logN}{log(1/r)} \tag{12}$$

The dimension formula (12) can be applied to mathematical fractals, e.g. the Koch curve, Sierpinski triangle, etc. For instance, any segment of the Koch curve is composed of 4 segments each of which is scaled down by a factor $\frac{1}{3}$ from its parent. Hence its fractal dimension is $D_f = \frac{4}{3}$, which is about $D = 1,262$. The mathematical fractal is self-similar over an infinite range of scales, hence they have unlimited range of self-similarity.

3.1.2 | Fractal dimension of natural and random objects

There are two types of natural objects, subject to evaluation of their fractal dimension - objects which have a linear feature (e.g. coastlines) and objects which have surface features to be investigated (e.g. topographic surfaces, digital images). The natural like objects, which are generated by using iterative by stochastic procedure will be addressed here as random fractal objects.

The natural objects (land boundaries and surfaces) are not strictly self-similar because they have a limited range of self-similarity. They are considered as *statistically* self-similar [2], [17]. Hence the length of the coastline $L(r)$ varies *on average* as $1/r^{D_f}$, Voss [17].

$$L(r) \propto r\frac{1}{r^{D_f}} = \frac{1}{r^{(D_f-1)}} \tag{13}$$

where r is the size of the ruler, D_f is the fractal dimension.

One can distinguish between the fractal and non-fractal objects by observing the relationship between the object (fractal) size and the magnification factor. In non-fractal objects, there is not observable dependence between the object size and value of the magnification factor. If one plots the *log(fractal's size)* against *log(magnification factor)* of natural or mathematical fractal object, one should get a straight line. The slope of this line is a measure for the *fractality* of the object. The slope for the non-fractal objects should be zero. The fractal dimension as a basic measure for the fractality can be determined from the slope of this line.

This idea has been utilised by Richardson[8] and Mandelbrot[11] (see also Voss[17]), who derived the tools and methodology for evaluation of the dimension of *non-natural* fractals, known as Richardson-Mandelbrot (R-M plot).

Richardson in his empirical studies[8] showed the dependence between the scale the cartographic maps and the lengths of the political borders. He determined the rate of increase by calculating the slope of the regression line of the log-log plot of the length of the line against the size of the measurement instrument. Mandelbrot[2] reported the relationship between the fractal dimension and the slope of the Richardson plot. Namely

$$D_f = 1 - b, \tag{14}$$

for lines and

$$D_f = 2 - b, \tag{15}$$

for surfaces[9] and[17]. The variable b is the slope of the regression line, created from the points in the R-M scatter plot. Further we will designate the regression line trough the points in the R-M plot as *RM line*.

Different algorithms are applied to construct the R-M plot and to estimate the fractal dimension of natural objects. One can summarize them with the following steps:

- A sequence of step sizes of r, is applied to measure the quantity which represents the size of the object $L(r)$ (e.g. length or surface);
- A log-log R-M scatter plot $log(L(r))$ against $log(r)$ is constructed;
- A least-squares RM line trough the points of the R-M scatter plot is constructed;
- The slope b of the line is considered to be an estimation of the fractal dimension D_f, see (14), (15)

We will consider separately the two stages of determination of the slope (fractal dimension) - construction of the scatter plot and construction of the regression line.

R-M scatter plot

One of the basic procedures for determination of the parameters r and $L(r)$ to be plotted on R-M plot, is the divider method[4]. The method can be implemented by "walking" the divider along the line and record the number of steps to cover the line. One systematically increases the width of the divider and calculates $L(r)$ which corresponds to each size of r. Thus the stepping process is repeated, over a range of resolutions (sizes of r), until the size of the object is exhausted. Then on can determine the relation between the step sizes (divider's width) r and the respective measured line lengths $L(r)$.

A very popular method used both as method for line and area (see (14), (15)) D_f determination is the box-counting method[4]. The box-counting algorithm for evaluation of the fractal dimension of an object, having surface feature is one of the most popular methods to identify surface roughness. It is widely applied in characterizing digital images of surfaces[9],[10],[20],[3]. The method comprises the following steps:

- One covers the whole object with a single box. Then one divides the box into four quadrants (cells) and Counts the number of the box cells, occupied by elements of the object;
- Again divides each each subsequent quadrant in four sub-boxes and again counts the number of the occupied sub-boxes;
- One continues until the desired minimum box size (i.e. measuring accuracy) is reached;
- A $log - log$ plot of the number of occupied boxes $L(r)$ against the size of the boxes r is constructed (R-M plot);
- Considering that $L(r) \propto r^{-D_f}$, the fractal dimension D_f is determined from the slope of the RM line by applying the approach as discussed before ((13) - (15)).

A different approach for evaluation of the area of topographic surface is proposed by Clarke[9]. The author extends the idea of using walking divider, already applied in measuring lines. While the walking divider measures the length of a line by counting the steps of the divider, the presented method is based on counting the surface area of the top side of triangular prisms, which cover the topographic surface. The method makes use of a discrete representation of the elevations of the Earth's similar to a digital terrain model. Suppose one considers a gray-scale terrain picture each pixel represent the respective elevation of the terain. One considers a square of four pixels a, b, c and d as well the calculated average elevation e in the center. Four triangles can be drown a, b, e, b, c, e etc. The pixels $a - d$ are spaced at equal distances r. Each triangle is considered as the top surface of a triangular prism, where the values of the pixels represents the height of the prism's vertical edges. The surface of each triangular prism is calculated by using the Pythagoras theorem and the Heron formula for surface of triangle. The sum of the four triangles represents the whole surface of the square prism, made from the four triangle prisms. The surface of the immage is calculated by summing the surfaces of all square prisms. To calculate the fractal dimension one consider the log-log R-M plot of the total surface of the image against the size of the squares prisms.

As pointed in[21] a critical step for the prism method is the calculation of the surface areas of the prisms. The method calculates the surface, based on the surfaces of the triangles. But each triangle is formed from the elevation (measured) of the corners $a-d$ and the value at the center e (interpolated). Another important peculiarity of the prism method is the fact, that it presumes finite decreasing (up to one) the side of the squares, which is the scaling factor r.

De Santis at all.[22] considered two variants of the prism method - TPM2 and TPM4[9]. In TPM2, the top surface of a square prism is separated on two triangles, determined by the vertical edges of the prism A1 . No middle averaged elevation is calculated, The TPM4 method splits the top surface of the prism on four triangles, determined by the vertical edges of the prism and also by the point which is an average of the corner points.

The TPM2 method was criticized in[22] for providing tendency of overestimation of the surface, in comparison with TPM4. But one can observe that it was used on synthetic objects - cube and half sphere[22]. These Euclidean surfaces are smooth and the value of the averaged middle point (in case of TPM4) does not differs significantly from the values at the prism vertices.

The averaging of the values at the corners to interpolate the value in the middle point was critisized[21], considering the case where sharp variations exist in the neighboring pixels. This can be expected especially for the matrices with random origin (e.g. the weight matrices of the Neural Networks).

The TPM4[9] was developed for calculation of the surface area for topographic surfaces, where generally the roughness of the surface is much lower then the surface created by the weight matrix. The origin of the neighboring points (pixels) of topographic surface are nature forces, which common for the neighboring areas, which would result in relatively gradual variation of the elevations. The general rule with some exceptions though, that the neighboring pixels will not have extreme differences.

In the same time the values of the cells of the weight matrix, started as randomly generated. So, it is expected that their values will be non-correlated to each other and will have extreme differences. It is expected as a general rule that the neighboring cells will have extremely different values.

This is the reason to prefer the TPM2 method (following[22]) which utilizes only the values of the corner elevations, over TPM4, which uses also interpolated averaged middle point.

TPM2 was applied by[23] in terrain description, which, together with its simplicity encourage us to use it as the method for evaluation of the fractal surface of the MLP weights surface matrix.

Very important subject is the method for determining of the *local window* - the size of the bottom surface of the square prism. In fact this is closely related with the size of reduction factor r, to be used in the $R - M$ plots. In their work *divisor-step* method,[10] Ju and Lam compared four algorithms for determining the local window, applied in the triangular prism method and propose *divisor-step* method. They claim that the *divisor-step* method guarantees 100% effective coverage of $n \times n$ matrix, where n is an odd number. Hence in this work as a method for determining the size of the local window, we employed the *divisor-step* method,[10]. As the rest of the methods, the *divisor-step* method requires square matrix to be .

The object, to evaluate its fractal dimension, discussed in this article, is the weight matrix $W^{H_a H_b}$, paired with layer H_b, which elements $w_{ij}^{H_a H_b}$ are associated with the arcs, connecting the respective nodes of layer H_a with these of layer H_b.

$$W^{H_a H_b} = \begin{bmatrix} \vdots & \\ \dots & w_{ij}^{H_a H_b} \quad \dots \\ \vdots & \end{bmatrix}^{H_a H_b} \qquad i = 1, n; j = 1, n. \tag{16}$$

To preserve the analogy, we can regard the elements $w_{ij}^{H_a H_b}$ of $W^{H_a H_b}$, as pixels of a digital gray-scale image, which values are the weights (in the range $[0; 255]$), recalculated to vary in the range $[0; 255]$. The scaling factor will not be minimized any further than the size of single pixel.

Regression line

As mentioned above, Richardson[8], Mandelbrot[2], see also[24], introduced the concept of fractal dimension. The tool for evaluation the fractal dimension is a plot (called here *R-M plot*), where the measured characteristics of the objects (length of a coastline, topographic surface) at different values of the measuring interval are plotted in log-log scale. Then, by using the points, a regression line (called here *RM line*) is constructed. The slope b of the RM line can be utilized further to calculate the fractal dimension D_f of the object by using the equation $D_f = 2 - b$.

In this article we discuss the fractality of the NN and especially one of its key components - the weight matrices W.

Let us consider the assertion that a fractal is a set, where the Hausdorff-Besicovitch dimension strictly exceeds the Euclidean topological dimension[11]. To regard an object as fractal and considering (14) and (15), therefore, one need to prove that the slope of the RM line significantly differs from zero. Hence from now on we will concentrate mainly on the value of the slope b, rather on D_f.

For the cases where the object is an artificially generated fractal (e.g. the Koch curve) all points in the $R - M$ plot, lays on a line. While for objects with natural or stochastic origin (e.g. coast line, financial market curve) the points are spread around the RM line with some variance. Hence for natural objects, the evaluation of the fractal dimension, should utilize some statistical approach. Here we will introduce the term fractality of the object, which will characterize the level the points of the R-M plot are close to a line.

In the process of creating the Richardson-Mandelbrot scatter plot, we collect a set of p points $(X_1, Z_1), (X_2, Z_2), \dots X_p, Z_p$, where

$$\begin{aligned} X_i &\equiv \log(r_i), \\ Z_i &\equiv \log S(r_i), \end{aligned} \quad i = 1, p \tag{17}$$

.

To create a regression line, corresponding to these points, we need to assume that there is a linear dependence of $\log S(r_i)$ on $\log(r_i)$. Hence to build a linear regression model

$$\hat{Z} = b_0 + bX \tag{18}$$

where the pairs $\hat{Z}, X$, correspond the points of the RM line, corresponding to the values of X. The coefficients b_0 and b are respectively the intercept term and is the regression coefficient. Extensive study on the theory and practice of the regression analysis, one can find in[25].

Considering (17) the RM line, proposed by Richardson and Mandelbrot can be expressed as

$$\log S(r) = b_0 + b \log(r) \tag{19}$$

Here b is the slope of the line, which will be used for calculating of the fractal dimension for line $D_l = 1 - b$ and surface $D_s = 2 - b$, respectively.

As a criterion for the quality of the regression model (19) we will use the coefficient of determination R^2. Also one can consider that $R = \sqrt{R^2}$ is the popular coefficient of multiple correlation.

The coefficient of determination R^2 is a measure for the correlation of the independent variable $X \equiv \log(r)$ on the dependent variable $\hat{Z} \equiv \log(S(r))$, which represents the RM line. R^2 evaluates the correlation of the dependent variable in our case $\log(Fractal\ surface)$ and the independent variable - in our case the step size $\log(step\ size)$. The closer the value of R^2 to 1, the bigger is the correlation between the measured values $r_i, i = 1, p$ and the predicted by the linear regression RM line values $\hat{Z}_i, i = 1, p$.

4 | FRACTALITY OF A NN WEIGHT MATRIX

As is was discussed already, an object can be considered as fractal, if it posses two important features - (statistical, if the object is natural) self similarity and fractal dimension, greater than the respective topological dimension.

4.1 | Statistical Selfsimilarity of the MLP

The Neural Networks (NN) are composite functions[26] and Appendix (B). The training process of each NN goes iteratively trough stages called *Epochs*. During each *Epoch* the whole training set is utilized to recalculate the weight coefficients of the NN. During this process the parameters, subject to recalculation are the weight matrices of the Neural Network. They are recursively adjusted to improve the value of the Loss function. We can postulate that the NN training process is a transformation procedure, where the weight matrices are gradually transformed from completely random, made of evenly generated random numbers set, to a set possessing the feature fractality. The weight matrices, starting for random matrices, gradually receives fractal properties.

Based on this and considering (1) and (6) and also considering that the Neural Network is a computational structure aimed to calculate and re-calculate the values of the elements of the weight matrices, which in turn have stochastic origin, one can conclude that the Neural Networks (fully connected MLP) are statistical self similar objects.

4.2 | Fractal dimension of a weight matrix

Let us consider a fully connected Neural Network (MLP) with follows the following architecture.

Input and output layers - I and O. Two hidden layers H_1 and H_2, having h_{H_1} and h_{H_2} nodes respectively, also $h_{H_1} = h_{H_2}$. Each hidden layer as well the output layer is paired with a weight matrix $W^{IJ}_{h_I h_J}$, where I and J are the designations of the layers, while h_I and h_J are the respective number of the nodes in the layers.

The detailed structure of the network is shown on (Figure 3):

- Input layer I
- weight matrix, $W^{IH_1}_{nh_{H_1}}$
- hidden layer, H_1
- weight matrix, $W^{H_1H_2}_{h_{H_1}h_{H_2}}$
- hidden layer, H_2
- weight matrix, $W^{H_2O}_{h_{H_2}m}$
- output layer, O

As it was discussed above, one of the main features which determine some object as a fractal is the condition $D_f > D_E$, where D_f is its fractal dimension, D_E is its topological dimension. The fractal dimension of an object having surface features, can be calculated by using $D_F = 2 - b$, where b is the slope of the RM line, constructed in the $R - M$ plot, discussed already. Apparently to evaluate the *fractality* of particular object, one has to demonstrate that b is negative and differs from zero.

The goal of this investigation was the fractality of the weight matrix, $W^{H_1H_2}_{h_{H_1}h_{H_2}}$, associated with the second hidden layer H_2. The reason to choose this matrix is the fact that due to equality of the number of the nodes in H_1 and H_2 it is square, which allowed us to apply the *divisor-step* method,[10], which will be detailed further. Also because the NN have two hidden layers H_1 and H_2, this is the only weight matrix, which is square.

During the investigation we have varied the number of the nodes of layers H_1 and H_2, keeping $h_{H_1} = h_{H_2}$. Each training was continued until $e = 1500$ *Epochs* were performed.

For the purposes of this investigation, the evaluation of the slope of the RM line goes trough two stages

- Training of the Neural network
- Creation of the RM line of $W^{H_1H_2}_{h_{H_1}h_{H_2}}$ of the already trained network

NN training For the purposes of this investigation we have employed the Fashion MNIST data base[27]. The training set of the data base contain $N = 60000$ samples with $n = 784$ features. Each sample was labeled with a vector $m = 10$ elements, associated with the 10 classes, the samples should be classified to.

The training set X_{Nn} is comprised of N samples with n features, while the set of m classes is designated as Y_m.

During the calculations the training set X will be processed as a single batch, so the input layer I, will be consisted of the whole training set X, the input layer in fact is the matrix X_{Nn}, while the output layer is the vector Y_m, hence $I \equiv X_{Nn}$, $O \equiv Y_m$.

The first *Epoch* starts with generation of a set of evenly distributed set of random numbers, which becomes the initial elements of the first weight matrix $W^I H_1$. Then the elements of the second weight matrix $W^{H_1H_2}_{h_{H_1}h_{H_2}}$ are calculated from $\gamma(X \times W^{IH_1})$.

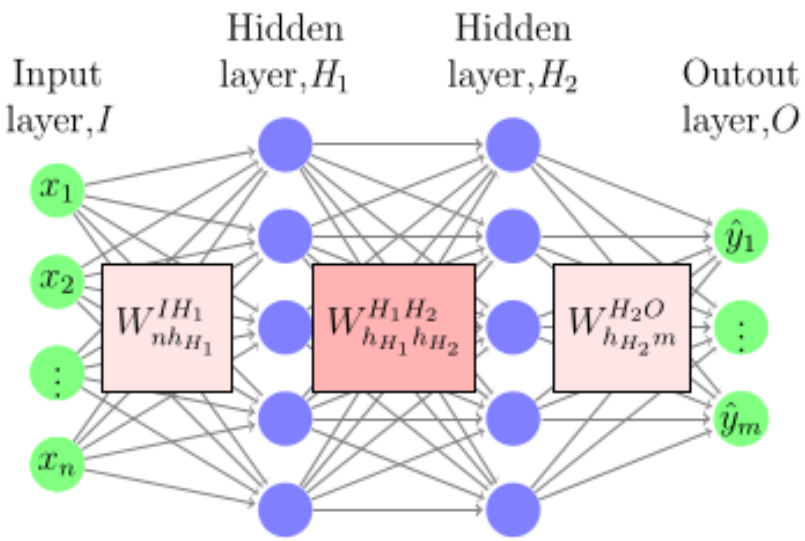


FIGURE 3 Fully connected Multi Layer Perceptron (MLP) with input (I), output (O) layers and with two hidden layers H_1 and H_2. The hidden layers H_1, H_2 and the output layer O are paired with weight matrices. The weight matrix $W^{H_1H_2}_{h_{H_1}h_{H_2}}$ is associated with the arcs, connecting layers H_1 and H_2 (superscripts) having h_{H_1} and h_{H_2} nodes respectively (subscripts).

$$W^{H_1H_2}_{h_{H_1}h_{H_2}} = \gamma(X \times W^{IH_1}) \tag{20}$$

where $\gamma(\cdot)$ is the activation function.

The elements of the next weight matrix W^{H_2O} are calculated by a similar way.

$$W^{H_2O}_{h_{H_2}m} = \gamma(W^{H_1H_2}_{h_{H_1}h_{H_2}}) \tag{21}$$

Finally the values of the predicted probability distributions are calculated

$$\hat{Y} = \gamma\left(W^{H_2O}_{h_{H_2}m}\right) \tag{22}$$

The Loss function (1), (5) is calculated and the values of the elements of the weight matrices are recalculated by back propagating the derivative of Λ (6). The Loss and Accuracy values are calculated to evaluate the performance of the neural network. The forward calculation and backward propagation, make an *Epoch* at the end of which a new initial weight matrix W^{IH_1} is calculated.

A new *Epoch* starts and a number of *e Epochs* is performed eventually.

After the last *Epoch is finished* one considers the NN as trained. Then the slope of the RM line of $W^{H_1H_2}_{h_{H_1}h_{H_2}}$ is evaluated.

Creation of the RM line of $W^{H_1H_2}_{h_{H_1}h_{H_2}}$

The method for computing the Hausdorff-Besicovitch dimension of a line (e.g. the coast of Britain) involves taking of a cartographic line and repeatedly measuring its length by using different resolution of the instrument. For instance one can take a divider and by walking it along the line while counting the number of steps necessary to reach the end. For each walking trail, along the line, performed with different size of the divider, the size of the divisor s is recorded, and in parallel also the number of steps, necessary to reach the end of the line, multiplied by the size of the divisor (which is the measured length of the line) - $L(s)$. A scatter plot of the log transforms of the pairs of these values is constructed. Next step will be to build a regression line trough the points and to record its slope b and and respectively the fractal dimension $D_f = 1 - b$.

As method for evaluation of D_f of the roughness of topographic surface, a direct analogy of the dividers method, was proposed by Clarke[9]. In its original method Clarke assumes a discrete representation of the earth's surface such as a digital terrain model. The terrain elevations are spaced at the intersections of the cells of a uniformly spaced square grid. The resolution of the grid r and the size of the map, where r will be increased to, defines the extremes of the scale variations of r. The largest square is

computed first from the length of the shortest side of the map, and the largest power of two smaller than the side is selected for the maximum side of computed squares. If particular size is desired (e.g. 256×256) the map area grid should be made one cell larger (257×257)[9].

Continuing with the line analogy, while the size of the divisor s represents its length$L(s)$, the size of the square with side r will be the surface of the top side of a prism$S'(r)$, built with the squire as bottom side. According to the Clarke's method, which we will address further as TPM4, the square prism, based on the square with side r is made of four triangle prisms. These four prisms share an edge in the center of the square. One takes the levations in foru pixels a, b, c and d. Then calculates the the interpolated average elevation (assigned to the center) e. To create each of these four prisms, one takes the values of the elevations at two adjacent pixels together with the center e. The height of each corner pixel is the pixel intensity value and also the height of the edge in the center. The surface of the top side is calculated by using the Pythagoras theorem and the Heron's formula. Then the surfaces of the four prism are summed up, which results in the top surface $S'(r)$ of the square prism. The evaluated in such a way quantity $S'(r)$ is equivalent of s for the line situation. To evaluate the surface equivalent of $L(s)$ one needs to sum up all top surfaces of the prisms $S(r)$, constructed with particular value of r. Hence the pairs $r/S(r)$ will be used to construct a a log-log scatter plot and to evaluate a regression line, which slope b will be used fo calculation of the fractal dimension ($D_f = 2 - b$) of the topographic surface. Extensive research on the *Prism method* and other methods, one can find in[9], [10], [3], [28], [23], etc.

To calculate the fractal dimension of a weight matrix $W^{H_1 H_2}_{h_{H_1} h_{H_2}}$, we will apply a variation of TPM4, called the TPM2 method,[22]. The TPM2 is based again on four adjacent pixels a, b, c and d, but instead of calculating the center edge and building four triangle prisms around it, one takes three adjacent pixels and by using analogical geometry, by using the Pythagoras theorem and Heron's formula calculates $S'(r)$ top surface of the so far constructed triangle prism. See Figure 4 and Figure A1 . Our goal here, again, will be to determine a set of pairs $r/S(r)$, to be used to construct the scatter plot (*R-M plot*) and based on it - the regression line *RM line*.

We need to determine a set p sizes of squares $\mathcal{R} = [r_1, r_2, \dots, p]$, then to cover the matrix with a grid made of these squares, with sizes $r_1, r_2, \dots, p$ and to calculate the set of surfaces $S(r_1), S(R_2), \dots, p$. The R-M plot will be constructed from the pairs $r_i/S(r_i)$.

The process of creating the R-M plot, starts with recalculating the elements of $W^{H_1 H_2}_{h_{H_1} h_{H_2}}$ from the range $[-1; 1]$, as they appeared after the training, to the range $[0; 255]$. This allow us to regard the weight matrix as two dimensional digitized gray-scale image, which will allow us to apply the whole technology[20], [3], [29], developed already to analyze such objects. Each of the cells of the weight matrix $W^{H_1 H_2}_{h_{H_1} h_{H_2}}$ matrix, by analogy with a digital image we will call a *pixel*. Hence the weight matrix will be represented as $h_{H_1} \times h_{H_2}$ sized gray-scale digital image.

The R-M plot is a log-log plot of the step size, r^2 and the respective total surface value. The slope of the regression line (RM line) is then calculated from the regression formula:

$$\widehat{Log(S(r))} = a + bLog(r^2) \tag{23}$$

where $S(r)$ is the sum of the top surfaces of the square prisms, constructed with step r (r^2 is the surface of the bottom side). Respectively a is the intercept coefficient, while b is the regression coefficient, which is also the slope of the RM line. We use the "hat" symbol $\widehat{(\cdot)}$ to distinguish between the measured values of $LogS(r)$ and the calculated by the equation (23) values $\widehat{Log(S(r))}$.

In the implementation of TPM2, we used the *divisor-step* method, where each step is the divisor of the size of the weight matrix h (also h is always odd number)[10].

The fractal surface $S(r)$ was calculated by using the Pythagoras theorem and the Heron formula[9]. The values of r are the sizes of the prism bases.

For example see Figure 4 let us consider the weight matrix paired with the hidden layer H_2, having $h_{H_2} = 11$ nodes. The divisors of $h_{H_2} - 1 = 10$ are 1, 2, 5. Hence we will utilize three values of r, 1,2 and 5 respectively. This will create three sets of square prisms with size, 1, 2 and 5. The weight matrix, covered with square prisms where $r = 1$ is presented on 5 , while the weight matrix, covered with prisms, where $r = 2$ is shown on 6 . The corners of the prisms, which corresponds to the vertical edges of the prisms are positioned in the respective cells

4.3 | Significance of R^2

To construct the R-M plot and the respective RM line, we start with generation of the set of log-transformed values of the sizes $\mathcal{R} = [r_1, r_2, \dots, p]$ and the corresponding fractal surfaces $S(r_i), i = 1, p$. Our subject of investigation is an object, which is

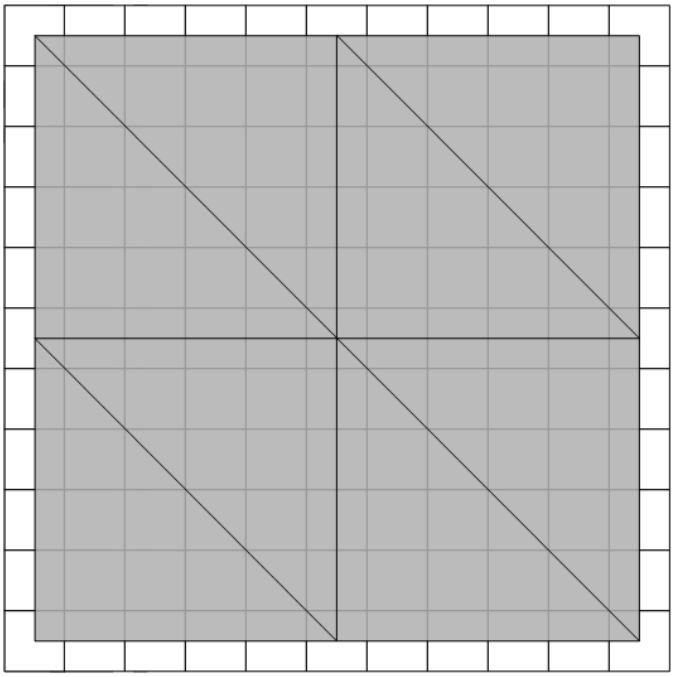


FIGURE 4 A matrix of 11×11 pixels, covered by 4 prisms, where $r = 5$. Each square prism is made of two triangle prisms

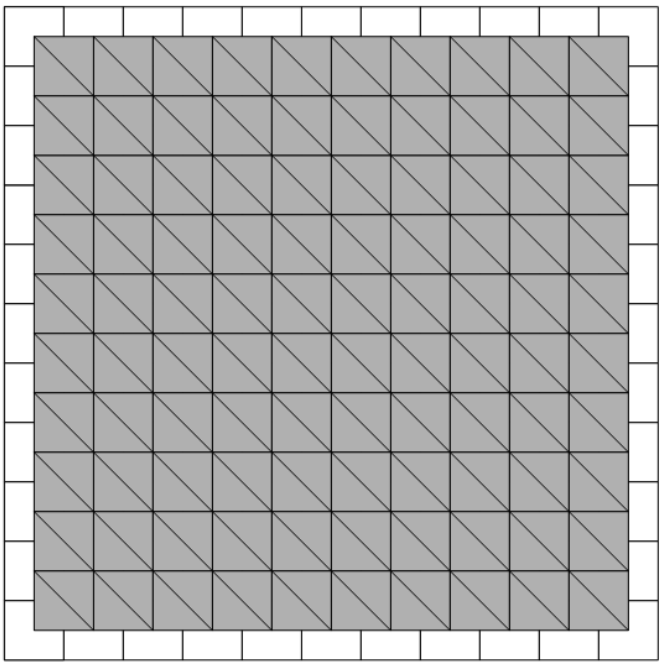


FIGURE 5 A matrix of 11×11 pixels, covered by 100 prisms, where $r = 1$. Each square prism is made of two triangle prisms

statistical self-similar, therefore the linear dependence if not an intrinsic feature. So we need to postulate that there exist a linear dependence between the values of surfaces $S(r_i), i = 1, p$ and the respective square sizes $r_i, i = 1, p$. Then, by using a metric like R^2 we can prove the level of self-similarity or the fractality of the object. Once we asume that there exist linear dependence, we can apply the regression analysis methodology[25,30], to evaluate the regression coefficients a, which also called intercept and b.

For simplicity we will express (23) as

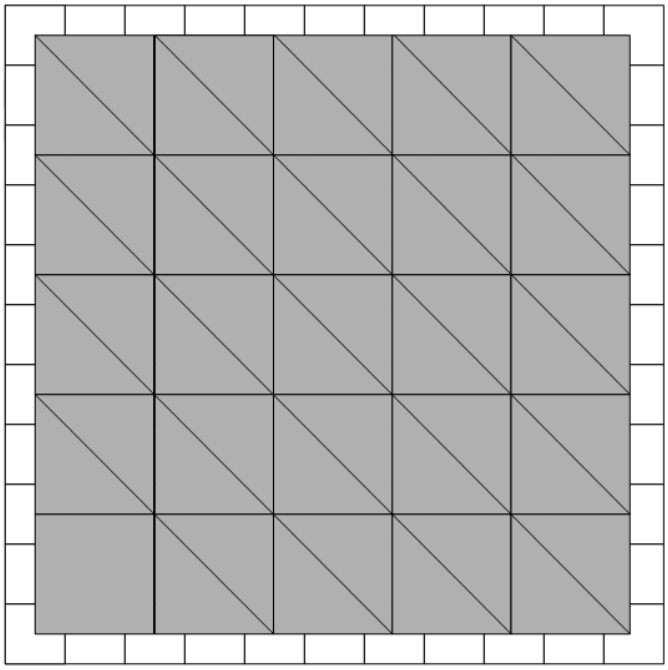


FIGURE 6 A matrix of 11×11 pixels, covered by 25 prisms, where $r = 2$. Each square prism is made of two triangle prisms

$$\hat{Z} = a + bP \tag{24}$$

where $\hat{Z} \equiv \widehat{Log(S(r))}$, $P \equiv Log(r^2)$. To evaluate the linearity of the set of pairs $[S(r_1)r_1; S(r_2)r_2; \ldots, S(r_p)r_p]$, hence to be able to construct a line, which the RM line, we will use the coefficient of determination R^2 [25].

The coefficient R^2, where $0 \leq R^2 \leq 1$ is a measure for the linearity of the relation between r_i and Z_i, where Z_i, $i = 1, p$ are random values with Normal distribution. If R^2 is close to zero, we can assume that there is no evidence for linear relationship. When R^2 is close to one, one can asserts that the relationship between r_i and Z_i is close to linear. In other words if the points lays on the regression line, the value of R^2 will be equal or very close to 1.0. Usually as a rule of thumb one accepts values higher than $R^2 \geq 0.8$.

One calculates the R^2 by using the following equation

$$R^2 = \sqrt{\frac{Q_R}{Q}} \tag{25}$$

where

$$Q_R = \sum_{u=1}^{p} \left(Z_u - \bar{Z}_u\right)^2 \tag{26}$$

$$Q = \sum_{u=1}^{p} \left(Z_u - \bar{Z}_u\right)^2 \tag{27}$$

The regression function of one variable, (24) which is used to determine the RM line, has two coefficients - the intercept a and the regression coefficient b. Theoretically one can construct a RM line by using only two points. It is known [25] that if the number of points is equal to the number of regression coefficients, the determination coefficient R^2 equals to one whatever is the distribution of the points. This is the reason why the common practice is, in parallel with the value of R^2 also its statistical significance to be considered. If there is a statistical assertion for the significance of R^2, then we could start drawing conclusions from its value.

Essentially we need to prove that there exists a linear dependence between the log transformed values of the square sizes $r_i, i = 1, p$ and fractal surfaces $S(r_i), i = 1, p$. Considering the random origin of the weights, one can conclude that $S(r_i), i = 1, p$ is randomly distributed variable. If we can claim that $S(r_i), i = 1, p$ is normally distributed variable, then we can apply R^2 as criterion. If we are not sure about the distribution of the Basically there are two approaches distribution free methods and methods

TABLE 1 Significance of the R^2

Number of nodes	r	R^2	Slope	$F_{(\alpha,\nu_1\nu_2)}$	F_{calc}	R^2 significant?
5	2	1	-0.80	∞	–	–
7	3	0.81	-0.36	161.45	8.53	No
9	3	0.97	-0.55	161.45	64.67	No
11	3	0.90	-0.49	161.45	18.00	No
13	5	0.83	-0.51	10.13	29.29	**Yes**
15	3	0.96	-0.46	161.45	48.00	No
17	4	0.83	-0.63	18.50	19.50	**Yes**
19	5	0.86	-0.56	10.13	36.86	**Yes**
21	5	0.73	-0.52	10.13	16.22	**Yes**
23	3	0.95	-0.51	161.45	38.00	No
25	7	0.87	-0.48	6.60	66.92	**Yes**

which presume normal distribution of the $S(r_i)$, we can apply distribution free methods, like the Kendall rank correlation coefficient. For the present article we will assume that the Central limit theorem holds. It will be a subject of future works to investigate the application of distribution free methods in evaluation of the correlation of the variables in the R-M plot.

To apply the R^2 based statistics, one have to assume that the involved variables are normally distributed random variables. One can note that the values of $S(r) \equiv Z$ depends on the values of the cells of the matrix W, which are evenly random distributed in the beginning of the process of training. Based on this one can conclude that are sums of the squares of Z (27), are normally distributed random quantities Q and Q_R, hence their ratio have χ^2 distribution. Therefore the value of

$$F_{calc} = \frac{R^2(p-k)}{(1-R^2)(k-1)} \tag{28}$$

has *Fischer* distribution with $\nu_1 = k-1$ and $\nu_2 = p-k$ degrees of freedom. As one can see the value of F_{calc} in (28) depends only of R^2, the number of the points p and $k = 2$[25],[30].

Apparently the number of the points in the R-M plot is closely related with the significance of the R^2. Based on this one can establish a simple rule for choosing the minimal number of points p, which will provide more confidence in the value of R^2. In case that we can choose the number of points p, we can apply the following procedure based on[30]:

- Choose number of points, p;
- Calculate F_{calc}, (28);
- Choose confidence level $\alpha = 0.05$, providing 95% confidence; Take the critical value of the Fisher, $F(\alpha, \nu_1, \nu_2)$
- If $F_{calc} > F(\alpha, \nu_1, \nu_2)$, then the R^2 is significant;
- If $F_{calc} < F(\alpha, \nu_1, \nu_2)$, then the R^2 coefficient is non significant, hence there is no evidence for linear relationship between dependent and independent variables.

If the determination coefficient R^2 appears to be a non significant, one should consider increasing of the number of points used for the estimation of the RM line befo re to start to draw conclusions, based on the value of R^2

To illustrate this approach, we investigated the significance of the R^2 over a series of weight matrix sizes, see Table 1 . A Neural Network with two hidden layers, similar to the one represented on Figure 3 , was constructed. The NN was applied for the classification task over the Fashion MNIST data set[27]. Subject to the investigation was the fractality of the weight matrix $W^{H_1H_2}_{h_{H_1}h_{H_2}}$. The task is to determine the fractal dimension of the weight matrix. To accomplish this, we need to calculate the slope of the RM line. In the same time need to consider only RM lines, where the assumption for linearity of the dependence between $Log(S(r))$ and $Log(r^2)$ is significant - hence the R^2 coefficient is significant.

We know that the number of points in the R-M plot depends on the number of nodes. It is related with the number of the divisors of the nodes. So if we take NN with many nodes, most probably we will have many points in the R-M plot. But the

large number of nodes will require more calculation resources. Therefore a balance between the number of nodes (calculation resources) and the number of points in the R-M plot (significance of R^2) should be established.

The weight matrices corresponding to $h_{H_2} = [5, \dots, 25]$ nodes in H_2 hidden layer were considered. The results are shown on Table 1 , where the significance of R^2 at different number of nodes (r) is presented. The first row, shows $R^2 = 1.00$, which the best possible value. But this value is also 100% unreliable. Despite that theoretically one can estimate a regression line having two coefficients with two points (r=2), one can not trust to the value of R^2.

Further in the table one can observe that R^2 is significant at five, among the eleven investigated cases, namely at 13,17,19,21 and 25 nodes. In all of these cases the number of the divisors (e.g. points in the R-M plot) where 4, 5 and 7. Considering that we need also to choose the minimum number of nodes (to save calculation resources), apparently the case with 13 nodes is the best one. Also one can observe that the value of the calculated F criterion is 29.29, which is almost three times higher than the $F_t(\alpha = 0.05, \nu_1, \nu_2)$. This is an additional condition to be assured as recommended[25]. The bigger the ratio $F/F_t(\alpha = 0.05, \nu_1, \nu_2)$ the more reliable statistically are the conclusions.

It worth to be noted here that the fact that the determination coefficient is statistically significant just provides confidence on its value. We can trust to it with 95% confidence. In the same time its value of $R^2 = 0.83$, which is rather mediocre value. This shows not very strong linear dependence in the R-M plot. Therefore we have to proceed with a caution with the value of slope and fractal dimension of this particular object.

Considering that the initial state of the Neural Network parameters is random, one could consider further attempts to generate weight matrix with 13 nodes, which possibly provide better value of R^2. As a general rule, which was illustrated here, in evaluation of the slope (fractal dimension), of some object (in our case the NN weight matrix) by using the Richardson-Mandelbrot plot, one should consider only these regression lines, which have coefficient of correlation which is both large (close to 1.0) and statistically significant.

5 | UTILIZATION OF THE FRACTALITY OF THE NN

We can utilize this and to use the value pf R^2 as a measure for the fractal nature of some object and particularly of the weight matrix $W^{H_1H_2}_{h_{H_1}h_{H_2}}$.

To monitor the development of the linearity feature of the object (the weight matrix), which starts from random state and gradually receives fractality state, we performed a training of the NN (Figure 3) of 1500 Epochs. At the end of each Epoch we have calculated the RM line and its R^2. The same MNIST Fashion classification task was tackled.

The results are demonstrated on Figure 7 . One can see that after several fluctuations, after completing 350-400 Epochs the value of R^2 settles down at levels of $R^2 > 0.95$. To compare, one can see that the values of R^2 behaves randomly in quite wide range.

One can conclude that with the progress of the NN training the points of the R-M plot tend to concentrate around a line, with quite high linearity.

By examining further the same Figure 7 , one can observe a typical behavior of the determination coefficient R^2 related with the number of the training epochs. On can see that in the begging, during the first appr. 200 epochs, the R^2 value demonstrates significant volatility, starting from zero (complete random $W^{H_1H_2}_{h_{H_1}h_{H_2}}$ matrix) continuing with large variance. One can observe that at about 400 Epochs the R^2 settles down to steady level, which remains until the end of the training, which in this case is 1500 epochs.The value of R^2 after 500 epochs reaches values above 0.95, which is also a statistically significant value. One can make some important conclusion here: **During the training of the Neural Network, the matrix $W^{H_1H_2}_{h_{H_1}h_{H_2}}$ which is also subject to recursive recalculation, acquires fractality, provided that the value $R^2 > 0.95$ and R^2 is statistically significant.** A similar to R^2 behavior one can observe on 8 . Which is not surprising, considering the relationship between the slope b of the RM line and R^2, see (29),[25]:

$$b = \frac{\sum(Z_i - \bar{Z})}{\sum(X_i - \bar{X})}\sqrt{R^2} \tag{29}$$

Therefore one can considering that $b \propto R^2$ one can use b and R^2, interchangeably, as a measure for the fractality of the weight matrix.

One can observe from 9 that the Loss function and Accuracy function reach asymptotically a stable values at approximately the same number of *Epochs* as the RM line slope (b). Therefore one can utilize the fractality and can use the slope as a measure

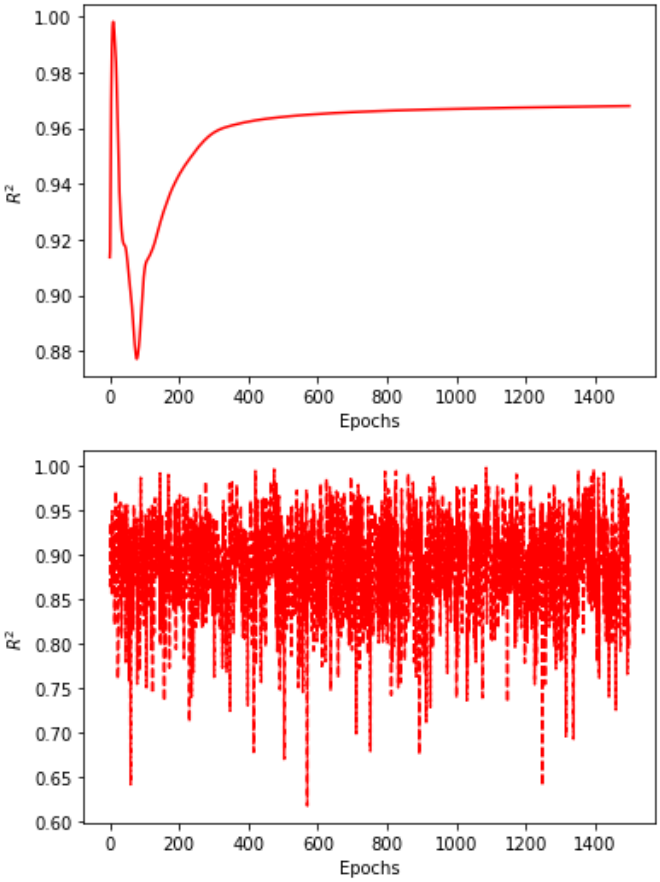


FIGURE 7 R^2 behavior of random matrices (right) and matrix during the process of training (left)

of the gained performance of the Neural Network. One can make another conclusion: **By observing the behavior of the slope of the RM line of the weight matrix, one can decide when to stop the training process of a MLP type of Neural Network.** Another significant resource engaging factor is the number of the nodes of the layer, associated with the weight matrix. As long the NN is fully connected each node is associated with weight, which is element of the weight matrix W. From Fig 10 one can observe the results of an investigation of the dependence of the values of the Slope, Loss and Accuracy from the number of nodes in the layer, paired with the weight function. Under this study the following range of nodes $[5-291]$ was investigated. At each number of nodes, the assumed number of 1500 *Epochs* was performed.

On the Figure one can see three major areas of behavior of the values of the Slope, Loss and Accuracy, which almost coincide to each other. At values ranged $[11-51]$ nodes the values of Loss and Accuracy steadily fluctuate around the best values. *Accuracy* at values above 0.95, while the *Loss* fluctuates at values between 0.5 and 0.8.

Another interesting range of number of nodes is at above 70 nodes, until the end, at 291 nodes. One can see almost straight line at *Accuracy* at levels close to 0.11 and *Loss* at values 2.3. These values can be explained with over-fitting of the model constructed by the Neural Network. The overfitting happens at too many nodes, hence if one constraints them-self at moderate number of nodes, where the slope of the RM line is also at moderate levels (e.g. up to appr. 50 nodes) one would save resources at better performance of the NN.

Also there is a transitional range of nodes between 50 and 70 nodes, where the performance of the Neural Network gradually decreases. Which seems logically as the level of over-fitting of the model is expected to accumulate, rather to appear suddenly.

There are some single picks within the transitional range of nodes, where some good performance appears. They can be considered as local extrema, caused by the random origin of the weight matrix. The variance of the *Slope* and *Loss* also is due to randomness of the weight matrix. One can see also that this fluctuation is rather low, compared with the fluctuation in the transitional range.

Based on the above mentioned observations, we can make the following Conjectures.

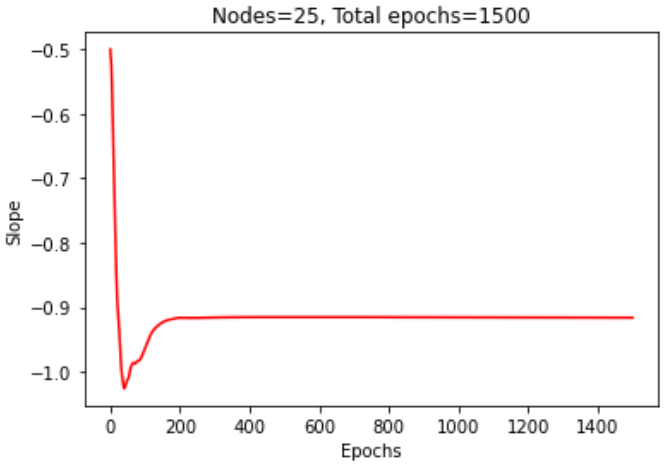


FIGURE 8 The behavior of the slope of the Richardson-Mandelbrot line during the process of training. The value of the slope settles down at about 200 epochs.

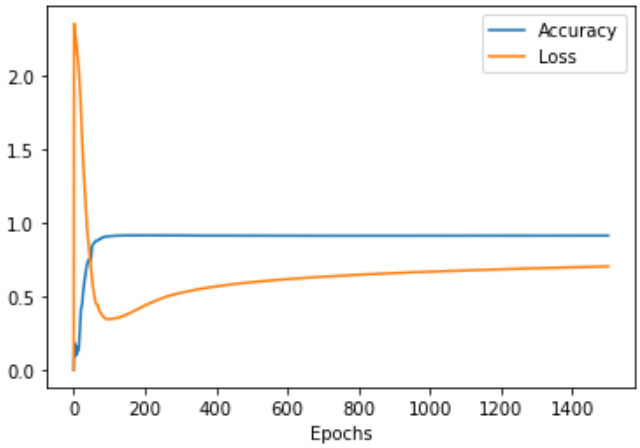


FIGURE 9 The behavior of the Accuracy and Loss values during the process of training. The Accuracy and Loss values settles down at about 400 epochs (similarly to the Slope value). Hence by observing the behavior or the slope or R^2, (see (29)) one can decide when to stop the training.

Conjecture 1. The composite nature of the Neural Network and the fractal dimension of the weight matrix $W_{h_I h_J}^{H_I H_J}$ demonstrate its fractality. This can be considered as a basis for the fractality of the Neural Network.

Conjecture 2. The fractality exhibits itself in the course of training of the NN.

Conjecture 3. The fractality exhibits itself at some number of iterations (*Epochs*), provided that the minimum number of nodes is provided.

Conjecture 4. Once the fractality state is achieved, the neither the increase of the number of *Epochs* nor the number of nodes will not affect either positively or negatively the predicting performance of the Neural Network.

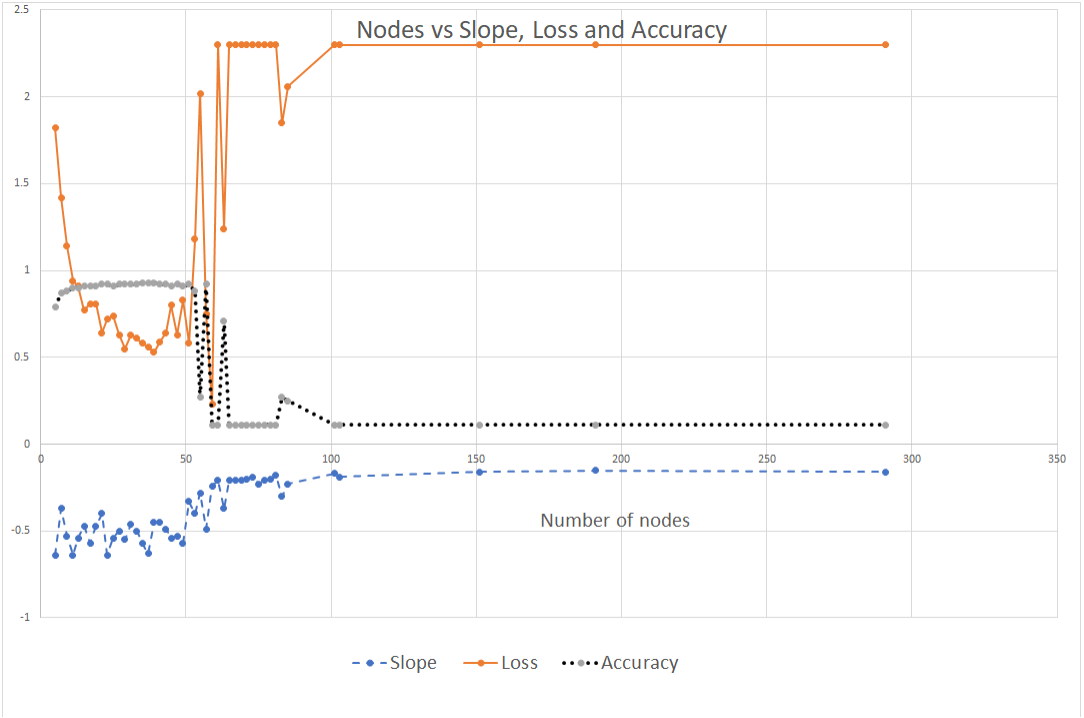


FIGURE 10

6 | CONCLUSION

The Multi-level perceptron (MLP), is a subject of the recursive process of training of its parameters, namely the weights of associated with the arcs connecting the nodes of the layers. This recursive process and the fact that the Neural Network (NN) is a composite function imply the fractal nature of the MLP. The estimated fractal dimension appears in the range $2.2 - 2.5$. This was illustrated with the evolution undergone by the Richardson-Mandlebrot[8,2] plots, constructed to evaluate the fractal dimension of a weight matrix. The plots were created by using of a variation of the Triangle Prism Method[9].

The fractal nature of the MLP was utilized as a method for forecast of the prediction quality of the NN and hence minimizing the number of the training set cycles (epochs) to be performed without losing assurance or loss values. Also it was demonstrated similar connection between the number of the nodes and the prediction performance (including overfitting) in terms of Loss and Accuracy with the slope of the Richardson-Mandlebrot regression line.

Some statistically rigorous rules for the determining of the slope in the Richardson-Mandlebrot regression line was demonstrated.

7 | ACKNOWLEDGMENTS

The software used[31,32,33,34,35], the used Neural Networks code[36].

This paper is supported by the National Scientific Program "Information and Communication Technologies for a Single Digital Market in Science, Education and Security (ICTinSES)", financed by the Ministry of Education and Science of Bulgaria.

Author contributions

This is an author contribution text. This is an author contribution text. This is an author contribution text. This is an author contribution text. This is an author contribution text.

Financial disclosure

None reported.

Conflict of interest

The authors declare no potential conflict of interests.

How to cite this article: Stoyanov K., J. Hristov (2021), Utilization of the fractal dimension metric in training of dense Neural Networks, *Math. Meth. Appl. Sci.*, *2021;00:1–6*.

APPENDIX

A EVALUATION OF THE UPPER SURFACE OF A TRIANGULAR PRISM BY THE TWO TRIANGLE PRISMS METHOD - TPM2

We construct a square prism by taking four adjacent elements $w^I_{11}J, w^I_{12}J, w^I_{21}J, w^I_{22}J$ of the weight matrix associated with the connections between the I^{th} and J^{th} hidden layers. To evaluate the surface are of the top side of the square prism, we will construct two triangle prisms, having common face. See Figure (A1), where the prisms share the face D, B, F, H. The basis of the prisms is a square with size length equals to r, which is step size, used in the evaluation of the fractal dimension. Namely $AB = BC = CD = DA = r, r = 1, \dots$.

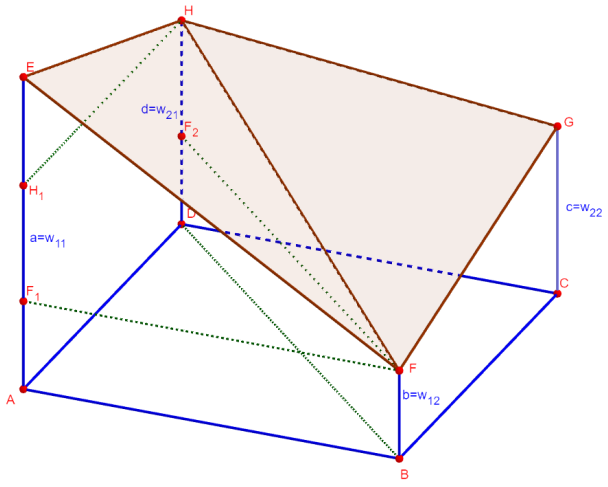


FIGURE A1 A square prism is formed of two triangular prisms with basis ABD and BCD. The surface of the pixel is the shaded area $EFGH$, representing its top base .

The base of the prism is at points A, B, C, D. The heights at each of base corners a, b, c, d are equal to the values in the cells of the matrix W^{IJ}, $w^{IJ}_{11}, w^{IJ}_{12}, w^{IJ}_{21}, w^{IJ}_{22}$, respectively. Te top base of the prism is E, F, G, H. Further for simplicity we will omit the IJ superscript.

We will start with calculating the area of one of the triangles, which forms the top base of the prism - E, H, F. We need to find the sides of this triangle and then by using the Heron's formula to calculate its surface area.

The triangle E, F, H forms the top base of the prism with bottom basis A, B, D. We know already that $AB = AD = j$, respectively $F_1F = AB = j$. So $EF_1 = EA - F_1A$, but $F_1A = EA - FB = w_{11} - w_{12}$, so $EF_1 = EA - FB$ or $EF_1 = w_{11} - w_{12}$.

Then we can calculate the length of EF by using Pythagoras theorem, where

$$EF = \sqrt{(F_1F)^2 + (EF_1)^2}, \tag{A1}$$

we receive

$$EF = \sqrt{j^2 + (w_{11} - w_{12})^2}, j = 1, \ldots \tag{A2}$$

Similarly the length of EH is $EH = \sqrt{j^2 + (w_{11} - w_{21})^2}$.

In determining the length of HF we consider the fact that $BD = FF_2 = j^2\sqrt{2}$, so

$$FH = \sqrt{j^2\sqrt{2} + (w_{21} - w_{12})^2}, j = 1, \ldots \tag{A3}$$

Next by using the Heron's formula, the surface of the top base of the first prism is

$$S_{EFH} = \sqrt{\sigma(\sigma - EF)(\sigma - FH)(\sigma - HE)}, \quad \text{where} \quad \sigma = \frac{EF + FH + HE}{2} \tag{A4}$$

By using the values of the cells of the weight matrix, the surface are of the first triangle prism is

$$S_{EFH} = \sqrt{\sigma\left[\sigma - \left(\sqrt{j^2 + (w_{11} - w_{12})^2}\right)\right] \sqrt{\left[\sigma - \left(\sqrt{j^2\sqrt{2} + (w_{21} - w_{12})^2}\right)\right] \sqrt{\left[\sigma - \left(\sqrt{j^2 + (w_{11} - w_{21})^2}\right)\right]}}}, \quad j = 1, \ldots \tag{A5}$$

where

$$\sigma = 0.5\left(\sqrt{j^2 + (w_{11} - w_{12})^2} + \sqrt{j^2\sqrt{2} + (w_{21} - w_{12})^2} + \sqrt{j^2 + (w_{11} - w_{21})^2}\right), j = 1, \ldots \tag{A6}$$

In a similar way we can evaluate the surface area of the top base of the second triangle prism, where

$$S_{GFH} = \sqrt{\sigma\left[\sigma - \left(\sqrt{j^2 + (w_{22} - w_{12})^2}\right)\right]\left[\sigma - \left(\sqrt{j^2\sqrt{2} + (w_{21} - w_{12})^2}\right)\right]\left[\sigma - \left(\sqrt{j^2 + (w_{22} - w_{21})^2}\right)\right]}, \quad j = 1, \ldots \tag{A7}$$

where

$$\sigma = \frac{\sqrt{j^2 + (w_{22} - w_{12})^2} + \sqrt{j^2\sqrt{2} + (w_{21} - w_{12})^2} + \sqrt{j^2 + (w_{22} - w_{21})^2}}{2}, j = 1, \ldots \tag{A8}$$

Finally the surface of the pixel is

$$S_{EFGH} = S_{GFH} + S_{EFH}. \tag{A9}$$

It is expected some of the the triangles to be needle shaped, for instance when the values are $w_{11} = 0, w_{12} = 255, w_{21} = 255$. To provide numerical stability of the Heron formula in such cases one can use the numerically stable version of the formula, proposed by Kahan[37].

B COMPOSITE FUNCTION

To demonstrate this we will consider a

$$\begin{aligned}
\hat{y}_1 &= \hat{y}_0 \equiv \vec{x} \\
\hat{y}_2 &= \alpha\left(\hat{y}_1, W_1\right) \\
\hat{y}_3 &= \alpha\left(\hat{y}_2, W_2\right) \\
&\vdots \\
\hat{y}_{n-1} &= \alpha\left(\hat{y}_{n-2}, W_{n-2}\right) \\
\hat{y}_n &= \alpha\left(\hat{y}_{n-1}, W_{n-1}\right) \\
\hat{y}_n &= \alpha\left(\hat{y}_{n-1}, W_{n-1}\right) \\
\hat{y}_n &= \alpha\left(\alpha\left(\hat{y}_{n-2}, W_{n-2}\right), W_{n-1}\right) \\
\hat{y}_n &= \alpha\left(\alpha\left(\alpha\left(\hat{y}_{n-3}, W_{n-3},\right) W_{n-2}\right), W_{n-1}\right) \\
&\vdots \\
\hat{y}_n &= \alpha\left(\alpha\left(\alpha\left(\ldots, \alpha\left(\hat{y}_1, W_1,\right), \ldots, W_{n-3}\right), W_{n-2}\right), W_{n-1}\right)
\end{aligned} \tag{B10}$$

$$\begin{aligned}
\vec{\hat{y}}^{[n]} &= \phi\left(\ldots, \phi\left(\phi\left(\phi\left(\vec{x}, W^{[0,1]},\right), W^{[1,2]},\right), W^{[2,3]},\right), \ldots, W^{[n-1,n]}\right) \\
&\equiv \Phi(\vec{x}, W^{[n]})
\end{aligned} \tag{B11}$$

Where,

- $\vec{\hat{y}}^{[n]}$ - the vector of the predicted values of the response, calculated by a Neural networks with $[n]$ hidden layers
- $[n]$ - number of the hidden layers
- $W^{[i,j]}, i = j-1, j = 1, n$ - an arbitrary matrix of the weights over the arcs, connecting i^{th} and j^{th} hidden layer
- $\phi(\cdot)$ - an arbitrary activation function
- $\vec{x}$ - is the features vector.

- $\vec{\hat{y}}^{[n]}$ is the vector of the predicted values of the response, calculated by a Neural networks with $[n]$ hidden layers
- By $[n]$ we designate the number of the hidden layers
- $W^{[i,j]}, i = 1, n, j = 1, n+1$ is an arbitrary matrix of the weights over the edges, connecting i^{th} and j^{th} hidden layer
- $\phi(\cdot)$ is an arbitrary activation function
- $\vec{x}$ is the features vector.

$$\begin{aligned}
\hat{y}^{[0]} &\equiv \vec{x} \\
\hat{y}^{[1]} &= \phi\left(\hat{y}_1, W_1\right) \\
\hat{y}^{[2]} &= \phi\left(\hat{y}_2, W_2\right) \\
&\vdots \\
\hat{y}^{[n-1]} &= \phi\left(\hat{y}_n, W_n\right) \\
\hat{y}^{[n]} &= \phi\left(\hat{y}_{n+1}, W_{n+1}\right)
\end{aligned} \tag{B12}$$

$$\begin{aligned}
\hat{y}^{[n]} &= \phi\left(\hat{y}^{[n+1]}, W^{[n+1]}\right) \\
\hat{y}^{[n]} &= \phi\left(\phi\left(\hat{y}^{[n]}, W_n\right), W^{[n+1]}\right) \\
\hat{y}^{[n]} &= \phi\left(\phi\left(\phi\left(\hat{y}^{[n-1]}, W^{[n-1]},\right) W^{[n]}\right), W^{[n+1]}\right) \\
&\vdots \\
\hat{y}^{[n]} &= \phi\left(\phi\left(\phi\left(\ldots, \phi\left(\hat{y}^{[1]}, W^{[1]},\right), \ldots, W^{[n-1]}\right), W^{[n]}\right), W^{[n+1]}\right) \equiv \\
\hat{y}^{[n]} &= \phi\left(\phi\left(\phi\left(\ldots, \phi\left(\vec{x}, W^{[1]},\right), \ldots, W^{[n-1]}\right), W^{[n]}\right), W^{[n+1]}\right)
\end{aligned} \tag{B13}$$

For example if we have a Neural Network with one input layer, three hidden layers and one output layer, the formula becomes:

$$\hat{y}^{[3]} = \phi\left(\phi\left(\phi\left(\vec{x}, W^{[1]}\right), W^{[2]}\right), W^{[3]}\right)$$

C NOTATION

- X - $N \times n$ matrix of patterns
- Y - $N \times p$ matrix of the labels associated with the patterns
- $\hat{Y}$ - $N \times p$ matrix of predicted labels
- m - number of classes
- N - number of samples in X
- p - number of points used for construction of the RM line in the R-M plot;
- n - number of features in X
- e - number of *Epochs*
- l - number of hidden layers
- h_A - number of nodes per hidden layer A. All hidden layers have equal number of nodes, h
- $W^{AB}_{h_A h_B}$ - the matrix of the weights associated with the arcs connecting two arbitrary hidden layers A and B, each having respectively h_A and h_B nodes;

References

1. Clarke Keith C, Schweizer Diane M. Measuring the fractal dimension of natural surfaces using a robust fractal estimator. *Cartography and Geographic Information Systems.* 1991;18(1):37–47.
2. Mandelbrot Benoit. How long is the coast of Britain? Statistical self-similarity and fractional dimension. *science.* 1967;156(3775):636–638.
3. Nayak Soumya Ranjan, Mishra Jibitesh, Palai Gopinath. Analysing roughness of surface through fractal dimension: A review. *Image and vision computing.* 2019;89:21–34.
4. Klinkenberg Brian. A review of methods used to determine the fractal dimension of linear features. *Mathematical Geology.* 1994;26(1):23–46.
5. Lam Nina Siu-Ngan, Qiu Hong-lie, Quattrochi Dale A, Emerson Charles W. An evaluation of fractal methods for characterizing image complexity. *Cartography and Geographic Information Science.* 2002;29(1):25–35.
6. LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning. *nature.* 2015;521(7553):436–444.
7. Shrestha Ajay, Mahmood Ausif. Review of Deep Learning Algorithms and Architectures. *IEEE Access.* 2019;7:53040-53065.
8. Richardson Lewis F. The problem of contiguity: an appendix to statistics of deadly quarrels. *General systems yearbook.* 1961;6:139–187.
9. Clarke Keith C. Computation of the fractal dimension of topographic surfaces using the triangular prism surface area method. *Computers & Geosciences.* 1986;12(5):713–722.
10. Ju Wenxue, Lam Nina S-N. An improved algorithm for computing local fractal dimension using the triangular prism method. *Computers & geosciences.* 2009;35(6):1224–1233.
11. Mandelbrot Benoit B, Mandelbrot Benoit B. *The fractal geometry of nature*. WH freeman New York; 1982.
12. *Online Collins English Dictionary.* 2021.
13. Goodfellow Ian, Bengio Yoshua, Courville Aaron, Bengio Yoshua. *Deep learning*. MIT press Cambridge; 2016.
14. Murphy Kevin P. *Machine learning: a probabilistic perspective*. MIT press; 2012.

15. Springer Verlag GmbH European Mathematical Society. *Encyclopedia of Mathematics.* 2021.

16. LeCun Yann, Bottou Léon, Bengio Yoshua, Haffner Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE.* 1998;86(11):2278–2324.

17. Voss Richard F. Fractals in nature: from characterization to simulation. In: Springer 1988 (pp. 21–70).

18. Klonowski Wlodzimierz. Signal and image analysis using chaos theory and fractal geometry. *Machine Graphics and Vision.* 2000;9(1/2):403–432.

19. Hearn Donald, Baker M Pauline. *Computer graphics with C.* Pearson; 1997.

20. Kraft Roland, Kauer Josef. *Estimating the fractal dimension from digitized images.* 1995.

21. Sun Wanxiao. Three new implementations of the triangular prism method for computing the fractal dimension of remote sensing images. *Photogrammetric Engineering & Remote Sensing.* 2006;72(4):373–382.

22. De Santis Angelo, Fedi Maurizio, Quarta Tatiana. A revisitation of the triangular prism surface area method for estimating the fractal dimension of fractal surfaces. *Annals of Geophysics.* 1997;40(4).

23. Piech M Ann, Piech Kenneth R. Fingerprints and fractal terrain. *Mathematical geology.* 1990;22(4):457–485.

24. Goodchild Michael F. Fractals and the accuracy of geographical measures. *Journal of the International Association for Mathematical Geology.* 1980;12(2):85–98.

25. Draper Norman R, Smith Harry. *Applied regression analysis*. John Wiley & Sons; 3 ed.1998.

26. Caterini Anthony L, Chang Dong Eui. *Deep Neural Networks in a Mathematical Framework.* Springer; 2018.

27. Xiao Han, Rasul Kashif, Vollgraf Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747.* 2017;.

28. Peleg S, Naor J, Hartley R, Avnir D. Multiple Resolution Texture Analysis and Classification. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. Pami-6, no. 4. 1984;.

29. Falconer Kenneth J. *The geometry of fractal sets*. No. 85Cambridge university press; 1986.

30. Vuchkov Ivan, Stoyanov Stoyan. *Mathematical modelling and optimization of technological objects*. Technica; 1980.

31. Van Rossum Guido, Drake Fred L.. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace; 2009.

32. Pedregosa F. at all. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research.* 2011;12:2825–2830.

33. *Anaconda Software Distribution.* 2020.

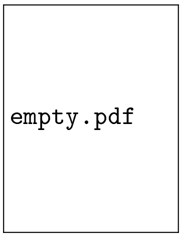
34. Harris Charles R. at all. Array programming with NumPy. *Nature.* 2020;585:357–362.

35. McKinney Wes, others . Data structures for statistical computing in python. In: :51–56Austin, TX; 2010.

36. Kinsley Harrison, Kukieła Daniel. *Neural Networks from Scratch $NNFS$*. 2021.

37. Kahan William. *Miscalculating area and angles of a needle-like triangle.* 2014.

AUTHOR BIOGRAPHY



Author Name. This is sample author biography text this is sample author biography text.