entropy

MDPI

Article

Rosenblatt's First Theorem and Frugality of Deep Learning

Alexander Kirdin [1,2], Sergey Sidorov [1,*] and Nikolai Zolotykh [1]

[1] Institute of Information Technologies, Mathematics and Mechanics, Lobachevsky State University, 603022 Nizhni Novgorod, Russia
[2] Institute for Computational Modelling, Russian Academy of Sciences, Siberian Branch, 660036 Krasnoyarsk, Russia
* Correspondence: sergey.sidorov@itmm.unn.ru

Abstract: The Rosenblatt's first theorem about the omnipotence of shallow networks states that elementary perceptrons can solve any classification problem if there are no discrepancies in the training set. Minsky and Papert considered elementary perceptrons with restrictions on the neural inputs: a bounded number of connections or a relatively small diameter of the receptive field for each neuron at the hidden layer. They proved that under these constraints, an elementary perceptron cannot solve some problems, such as the connectivity of input images or the parity of pixels in them. In this note, we demonstrated Rosenblatt's first theorem at work, showed how an elementary perceptron can solve a version of the travel maze problem, and analysed the complexity of that solution. We also constructed a deep network algorithm for the same problem. It is much more efficient. The shallow network uses an exponentially large number of neurons on the hidden layer (Rosenblatt's *A*-elements), whereas for the deep network, the second-order polynomial complexity is sufficient. We demonstrated that for the same complex problem, the deep network can be much smaller and reveal a heuristic behind this effect.

Keywords: complexity; classification; shallow network; elementary perceptron; deep network; travel maze problem

Citation: Kirdin, A.; Sidorov, S.; Zolotykh, N. Rosenblatt's First Theorem and Frugality of Deep Learning. *Entropy* **2022**, *24*, 1635. https://doi.org/10.3390/e24111635

Academic Editors: Alexander Gorban and Ivan Tyukin

Received: 29 August 2022
Accepted: 6 November 2022
Published: 10 November 2022

1. Introduction

Rosenblatt [1] studied elementary perceptrons (Figure 1). *A*- and *R*-elements are the classical linear threshold neurons. The *R*-element is trainable by the Rosenblatt algorithm, while the *A*-elements should represent a sufficient collection of features.

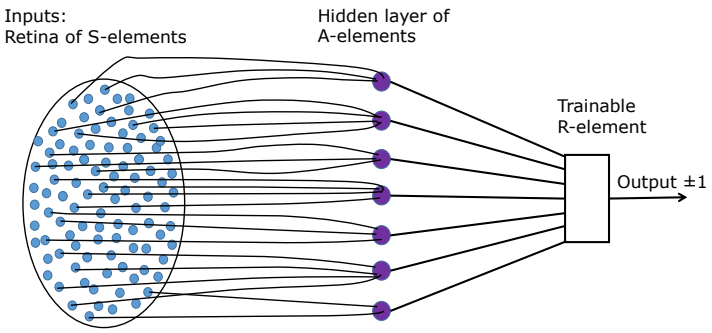

Figure 1. Rosenblatt's elementary perceptron (re-drawn from the Rosenblatt book [1]).

Rosenblatt assumed no restrictions on the choice of the *A*-elements. He proved that the elementary perceptrons can separate any two non-intersecting sets of binary images (Rosenblatt's first theorem in [1]). The proof was very simple. For each binary image x, we can create an *A*-element A_x that produces output 1 for this image and 0 for all others.

Indeed, let the input retina have n elements and $x = (x_1, \ldots, x_n)$ be a binary vector ($x_i = 0$ or 1) with k non-zero elements. The corresponding A-element A_x has input synapses with weights $w_i = 1/k$ if $x_i = 1$ and $w_i = -1/k$ if $x_i = 0$. For an arbitrary binary image y,

$$\sum w_i y_i \leq 1,$$

and this sum is equal to 1 if and only if $y = x$. The threshold for the A_x output can be selected as $1 - \frac{1}{2k}$. Thus,

$$OutA_x(y) = \begin{cases} 0, & \text{if } \sum w_i y_i < 1 - \frac{1}{2k}; \\ 1, & \text{if } \sum w_i y_i \geq 1 - \frac{1}{2k}. \end{cases} \quad (1)$$

The set of neurons A_x created for all binary vectors x transforms binary images into the vertexes of the standard simplex in $\mathbb{R}^{2^n}$ with coordinates $OutA_x(y)$ (1). Any two non-intersecting subsets of the standard simplex can be separated by a hyperplane. Therefore, there exists an R-element that separates them. According to the convergence theorem (Rosenblatt's fourth theorem in [1]), this R-element can be found by the perceptron learning algorithm (a simple relaxation method for solving of systems of linear inequalities).

Thus, Rosenblatt's first theorem is proven:

Theorem 1. *An elementary perceptron can separate any two non-intersecting sets of binary images.*

Of course, selection of the A-elements in the proposed form *for all* 2^n binary images is not necessary in the realistic applied classification problems. Sometimes, even an empty hidden layer can be used (the so-called linearly separable problems). Therefore, together with Rosenblatt's first theorem, we obtain a problem about the reasonable (if not optimal) selection of A-element. There are many frameworks for approaching this question, for example, the general feature selection algorithms: we generate (for example, randomly, or with some additional heuristics) a large set of A-elements that is sufficient for solving the classification problem, and then select the appropriate set of features using different methods. For the bibliography about feature selection, we refer to recent reviews [2–4].

The Minsky and Papert book *Perceptron* [5] was published seven years later than Rosenblatt's book. They started from the restricted percetrons and assumed that each A-element has a bounded receptive field (either by a pre-selected diameter or by the bounded number of inputs). Immediately, instead of Rosenblatt's omnipotence of unrestricted perceptrons, they found that the elementary perceptron with such restrictions cannot solve some problems, such as the connectivity of the image or the parity of the number of pixels in it.

Minsky and Papert proposed various definitions of restrictions. The first was the definition of the local order of a predicate. A predicate Ψ computed by an elementary perceptron is *conjunctively local of order k* if each A-element has only k inputs (depends only on k points) and

$$\Psi = \begin{cases} 1 & \text{if all outputs of } A\text{-elements } y_i = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Minsky and Papert studied flat binary image predicates. The first theorem about a limited perceptron stated that if S is a square lattice, then the *connectivity* of an image is not a conjuctively local predicate of any order. Then they published a much deeper theorem without the condition of conjunctive form but for the bounded number of inputs of A-elements: the only topologically invariant predicates of finite order are functions of the Euler characteristic of the image (Michael Paterson Theorem). Then they mentioned: "It seems intuitively clear that the abstract quality of connectedness cannot be captured by a perceptron of finite order because of its inherently serial character: one cannot conclude that a figure is connected by any simple order-independent combination of simple tests." The serial Turing machine serial algorithm for the connectivity test was presented with the evaluation of the necessary memory (the number of squares of the Turing tape): "For any ε

there is a 2-symbol Turing machine that can verify the connectedness of a figure X on any rectangular array S, using less than $(1+\varepsilon)\log_2|S|$ squares of tape". That is not very far from the deep multilayer perceptons with depth $\sim \log_2|S|$, but this step was not executed.

Minsky and Papert's results were generalized to more general metric spaces and graphs [6]. The heuristic behind these results is quite simple: if a human cannot solve the problem immediately at a glance and needs to apply some sequential operations such as counting pixels or following a tangled path, then this problem is not solvable by a restricted elementary perceptron.

At the same time, we can expect that the unrestricted elementary perceptron can solve this problem but at the cost of great (exponential?) complexity. Multilayer ("deep") networks are expected to solve these problems without an explosion of complexity. In that sense, deep networks should be simpler than shallow networks for the problems that cannot be solved by restricted elementary perceptrons and require (from humans) a combination of parallel and sequential actions.

As we can see, there is no contradiction between the omnipotence of unrestricted perceptrons and the limited abilities of perceptrons with limitation. Moreover, there is a clear heuristic that allows us to find the problems of "inherently serial character" (following the Minsky and Papert comment). A human cannot solve such problems with one glance and needs some serial actions, such as following tangled paths.

In this note, we demonstrate the relative simplicity of deep solvers on a version of the well-known travel maze problem (Figure 2). This geometric problem is closely related to the connectivity problem and has been used for benchmarking in various areas of machine learning (see, for example, [7]). The simplicity of the deep solution for this problem is not a miracle "because of its inherently serial character". A human solves it following the paths along their length L.

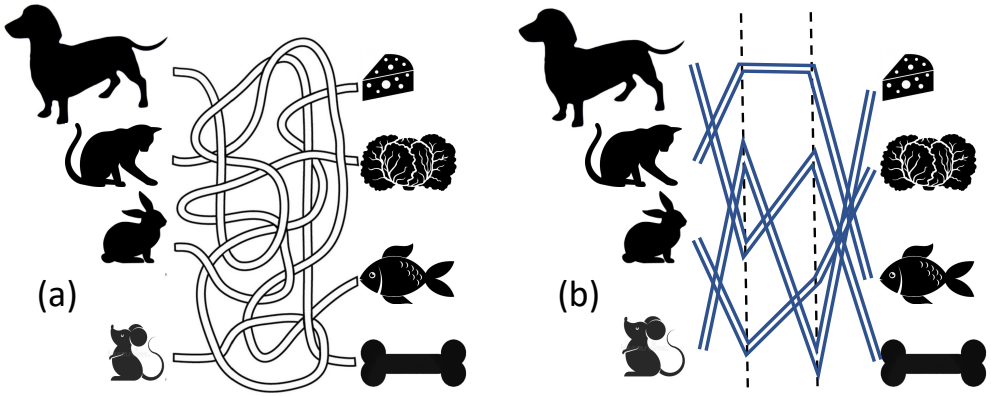

Figure 2. Have we chosen the right delicacies (**right**) for our guests (**left**)? (**a**) A prototype travel maze problem. (**b**) A simplified form of the problem with piece-wise linear paths for further formal description (Section 2). Complexity depends on the number of guests and the number of links in a path.

For formal analysis of the travel maze problem, we need to represent the paths on a discrete retina of S-elements (Figure 1). Then, to implement the logic of the proof of Rosenblatt's first theorem, each A-element should be an indicator element for a possible path. For each guest-delicacy pair in Figure 2a or Figure 2b, an elementary perceptron must be created that returns 1 if there is a path from this guest to this delicacy, and 0 if there are no such paths. Thus, n^2 elementary perceptrons should be created. We can easily combine them in a shallow network with n^2 outputs. To finalize the formal statement, we should specify the set of the possible paths. In our work, we select a very simple specification without loops, steps back or non-transversal intersections of paths (Figure 2b).

2. Formal Problem Statement

Consider the following problem. There are n people, each of whom owns a single object from the set $\{1, 2, \ldots, n\}$, and different people that own different objects. This correspondence between people and objects can be drawn as a diagram consisting of n broken lines, each of which contains L links (Figure 3). Each stage of the diagram consists of n links and can be encoded by a permutation or, equivalently, by a permutation matrix in a natural way. Namely, if an edge is drawn from node i to node $\pi(i)$ $(i = 1, 2, \ldots, n)$, then the permutation can be represented in the form

$$\pi = \begin{pmatrix} 1 & 2 & \cdots & n \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{pmatrix}$$

or by the permutation matrix $P = (p_{i,j})$, where

$$p_{i,j} = \begin{cases} 1, & \text{if } j = \pi(i) \\ 0, & \text{if } j \neq \pi(i). \end{cases}$$

If the permutation matrix X_i $(i = 1, 2, \ldots, L)$ corresponds to the i-th stage, then the product $X_1 \cdot X_2 \cdot \ldots \cdot X_L$ is the permutation matrix again, and it defines the correspondence "person–object". It is required to construct a shallow (fully connected) neural network that determines the correspondence "person–object" from the diagram.

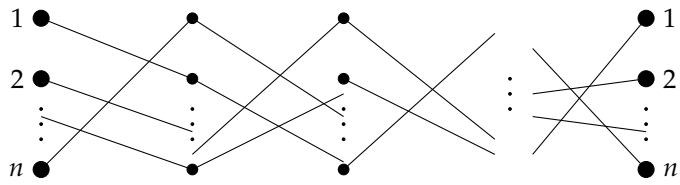

Figure 3. Game diagram with L stages (a formalized and simplified version of the travel maze problem).

We formally model a simplified version of the travel maze problem as follows. Each travel maze problem is an L-tuple $(f_1, \ldots, f_L)$ of bijective functions (permutations) from $\{1, \ldots, n\}$ to $\{1, \ldots, n\}$. Denote by $F(n, L)$ the set of such L-tuples. The composition $f_1 \cdot \ldots \cdot f_L$ of these functions from an L-tuple is also a bijective function $\{1, \ldots, n\}$ to $\{1, \ldots, n\}$ that one-to-one associates objects with persons. Thus, the set of all simplified versions of the travel maze problem is the set $F(n, L)$. In addition, the set $F(n, L)$ is a union of classes of L-tuples, each of which corresponds to the same product $f_1 \cdot \ldots \cdot f_L$.

When building the neural network for a travel maze problem, we want each neuron of the neural network to have Boolean values. In particular, each neuron of the next layer is a Boolean function of the neurons of the previous layer.

3. Shallow Neural Network Solution

Arrange all the $n!$ permutations $S_n = \{\pi_0, \pi_1, \ldots, \pi_{n!-1}\}$. Then, $M = \{P_0, P_1, \ldots, P_{n!-1}\}$ is the set of the corresponding $n \times n$ permutation matrices.

We denote the entries of the matrix P_k by $p^{(k)}_{i,j}$, where

$$p^{(k)}_{1,\pi_k(1)} = p^{(k)}_{2,\pi_k(2)} = \ldots = p^{(k)}_{n,\pi_k(n)} = 1,$$

and the other entries are equal to 0.

Let an L-tuple $(X_1, X_2, \ldots, X_L)$ $(X_i \in M$ for all $i = 1, \ldots, L)$ be an L-tuple (a word of length L) over the set of permutation matrices M. The number of such permutation matrices is $n!$, and the number of L-tuples with elements from M is $(n!)^L$. Consider all such words arranged by the lexicographical order. For each word $W_j = (X_1, X_2, \ldots, X_L)$

with the number j ($j = 0, 1, \ldots, (n!)^L - 1$), we assign the same number to the product $X_1 \cdot X_2 \cdot \ldots \cdot X_L = P_{t_j}$. The matrix P_{t_j} is also an $n \times n$ permutation matrix.

Entries of matrices $X_1, X_2, \ldots, X_L$ are inputs of the neural network (see Figure 4). Each input corresponds to an input neuron (S-element, Figure 1). An inner layer A-neuron y_j corresponds to the L-tuple $W_j = (X_1, X_2, \ldots, X_L)$ having the same number j. The neuron y_j should give the output signal 1 if the input vector is W_j and output 0 for all other $(n!)^L - 1$ possible input vectors. Other input vectors are impossible in our settings (Figure 4). Each matrix element of every permutation matrix X_i is either 0 or 1; therefore, the L-tuples star of output connections of the inner neuron can be coded as a 0–1 sequence, that is a vertex of the Ln^2-dimensional unit cube. (Apparently, there are more vertices than L-tuples of permutation matrices). This cube is a convex body, and each vertex can be separated from all other vertices by a linear functional. In particular, for each j, we can find a linear functional l_j such that $l_j(W_j) > 1/2$ and $l_j(W_k) < 1/2$ ($k \neq j, k, j = 1, \ldots, (n!)^L$). Here we, with some abuse of language, use the same notation for the tuple W_j and the correspondent vertex of the cube (a 0–1 sequence of the length Ln^2). Thus, each inner neuron y_j can be chosen in the form of the linear threshold element with the output signal (compare to (1) and Theorem 1):

$$y_j(W) = h(l_j(W) - 1/2),$$

where h is the Heaviside step function. We use for the output of the neuron y_j the same notation y_j.

The structural difference of the shallow network (Figure 4) for the travel maze problem from the elementary perceptron (Figure 1) is the number of neurons in the output layer. For the travel maze problem, the answer is the permutation matrix with n^2 0–1 elements. The inner layer neuron y_j detects the L-tuple of one-step permutation matrices $W_j = (X_1, X_2, \ldots, X_L)$. When this input vector is detected, y_i sends the output signal 1 to the output neurons connected with it. For all other input vectors, it keeps silent. The output neurons are simple linear adders. The output neurons z_{qr} are labelled by pairs of indexes, $q, r = 1, \ldots, n$. The matrix of outputs is the permutation matrix from the start to the end of the travel. The structure of the output connections of y_i is determined by the input L-tuple $W_j = (X_1, X_2, \ldots, X_L)$: the connection from y_j to z_{qr} has weight 1 if the corresponding entry $(P_{t_j})_{qr} = 1$ and is 0 if $(P_{t_j})_{qr} = 0$. (Recall that $P_{t_j} = X_1 \cdot X_2 \cdot \ldots \cdot X_L$).

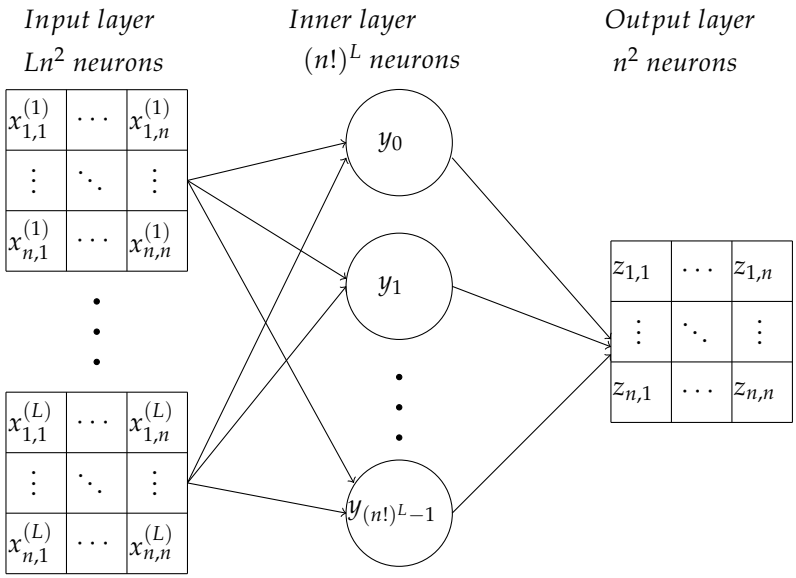

Figure 4. A shallow (fully connected) neural network for the travel maze problem (Figure 3). It differs from the classical elementary perceptron (Figure 1) by n^2 output neurons instead of one and can be considered as a union of n^2 elementary perceptrons with joint retina and hidden layer of A-elements.

Thus, the neuron y_j corresponds to our problem answer. Let us represent the network functioning in more detail with explicit algebraic presentations. All the inputs and outputs are Boolean (0–1) variables. We use the standard Boolean algebra notations. In particular, $\overline{x} = 1 - x$.

$$\begin{aligned}
y_0 &\longrightarrow \overbrace{P_0 \cdot \ldots \cdot P_0 \cdot P_0}^{L} = P_{t_0},\\
y_1 &\longrightarrow P_0 \cdot \ldots \cdot P_0 \cdot P_1 = P_{t_1},\\
&\ldots\\
y_{n!-1} &\longrightarrow P_0 \cdot \ldots \cdot P_0 \cdot P_{n!-1} = P_{t_{n!-1}},\\
y_{n!} &\longrightarrow P_0 \cdot \ldots \cdot P_1 \cdot P_0 = P_{t_{n!}},\\
&\ldots\\
y_{2n!-1} &\longrightarrow P_0 \cdot \ldots \cdot P_1 \cdot P_{n!-1} = P_{t_{2n!-1}},\\
&\ldots\\
y_{(n!)^L-1} &\longrightarrow P_{n!-1} \cdot \ldots \cdot P_{n!-1} \cdot P_{n!-1} = P_{t_{(n!)^L-1}}
\end{aligned}$$

Thus, if y_j is a neuron of the inner layer, then it corresponds to the product

$$P_{t_j} = P_{a_{j,L-1}} \cdot P_{a_{j,L-2}} \cdot \ldots \cdot P_{a_{j,1}} \cdot P_{a_{j,0}},$$

where

$$j = a_{j,L-1}(n!)^{L-1} + a_{j,L-2}(n!)^{L-2} + \ldots + a_{j,1}(n!) + a_{j,0}$$

is the expansion of j in the base $n!$.

We need

$$y_j(X_1, X_2, \ldots, X_L) = 1 \iff (X_1, X_2, \ldots, X_L) = (P_{a_{j,L-1}}, P_{a_{j,L-2}}, \ldots, P_{a_{j,0}}).$$

Denote

$$I_k = \{j : \ t_j = k\}, \ (k = 0, \ldots, n! - 1),$$

$$M_{ij} = \{k : \ \pi_k(i) = j\} = \{k : \ p_{i,j}^{(k)} = 1\}.$$

Note that $|M_{ij}| = (n-1)!$.

Each neuron y_j of the inner layer for $j \in I_k$ corresponds to the same product P_k:

$$y_j = x^{(1)}_{1,\pi_{a_{j,L-1}}(1)} \cdot x^{(1)}_{2,\pi_{a_{j,L-1}}(2)} \cdot \ldots \cdot x^{(1)}_{n,\pi_{a_{j,L-1}}(n)} \cdot x^{(2)}_{1,\pi_{a_{j,L-2}}(1)} \cdot x^{(2)}_{2,\pi_{a_{j,L-2}}(2)} \cdot \ldots \cdot x^{(2)}_{n,\pi_{a_{j,L-2}}(n)} \cdot$$

$$\ldots \cdot x^{(L)}_{1,\pi_{a_{j,0}}(1)} \cdot x^{(L)}_{2,\pi_{a_{j,0}}(2)} \ldots x^{(L)}_{n,\pi_{a_{j,0}}(n)} \cdot \prod_{(\alpha,\beta)\neq(i,\pi_{a_{j,s}}(i))} \overline{x^{(\gamma)}_{\alpha,\beta}}.$$

The third level neurons z_{ij} form the matrix $Z = (z_{ij})$ that is the answer to this problem:

$$z_{i,j} = \bigvee_{s \in \bigcup\limits_{k \in M_{ij}} I_k} y_s, \ (i, j = 1, \ldots, n). \tag{2}$$

Since $|M_{ij}| = (n-1)!$, $|I_k| = (n!)^{L-1}$, then the right-hand side in the equality (2) contains exactly $(n-1)! \cdot (n!)^{L-1}$ terms y_s.

Theorem 2. *For travel maze problems $F(n, L)$, there is a shallow neural network that has a depth of 3,*

$$(L+1)n^2 + (n!)^L$$

neurons and

$$(L+1)n^2(n!)^L$$

connections between them.

The constructed network memorizes products in all L-tuples of permutation matrices, recognizes the input L-tuple of permutations, and sends the product to the output.

Example 1. *Consider* $n = 2, L = 3$ *(Figure 5). Then*

$$\pi_0 = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}, \pi_1 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, P_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, P_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

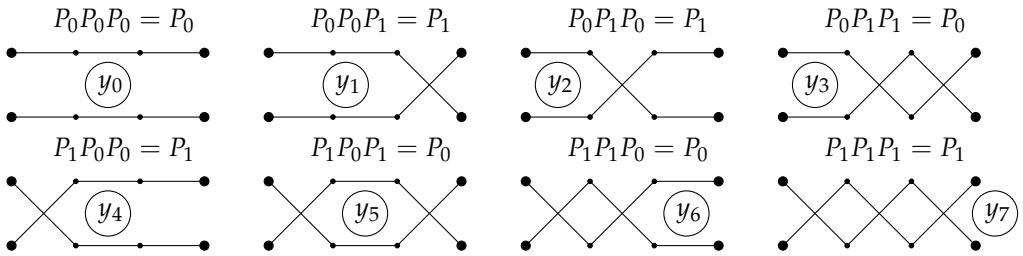

Figure 5. The case $n = 2, L = 3$.

$$X_1 = \begin{pmatrix} x_{11}^{(1)} & x_{12}^{(1)} \\ x_{21}^{(1)} & x_{22}^{(1)} \end{pmatrix}, X_2 = \begin{pmatrix} x_{11}^{(2)} & x_{12}^{(2)} \\ x_{21}^{(2)} & x_{22}^{(2)} \end{pmatrix}, X_3 = \begin{pmatrix} x_{11}^{(3)} & x_{12}^{(3)} \\ x_{21}^{(3)} & x_{22}^{(3)} \end{pmatrix}.$$

For example, we have $5 = 1 \cdot (2!)^2 + 0 \cdot (2!) + 1$ *for* $j = 5$*; therefore,*

$$y_5 = x_{1,\pi_1(1)}^{(1)} \cdot x_{2,\pi_1(2)}^{(1)} \cdot x_{1,\pi_0(1)}^{(2)} \cdot x_{2,\pi_0(2)}^{(2)} \cdot x_{1,\pi_1(1)}^{(3)} \cdot x_{2,\pi_1(2)}^{(3)} \cdot \prod_{(\alpha,\beta)\neq(i,\pi_{a_{j,s}}(i))} \overline{x_{\alpha,\beta}^{(\gamma)}} =$$

$$x_{1,2}^{(1)} \cdot x_{2,1}^{(1)} \cdot x_{1,1}^{(2)} \cdot x_{2,2}^{(2)} \cdot x_{1,2}^{(3)} \cdot x_{2,1}^{(3)} \cdot \overline{x_{1,1}^{(1)}} \cdot \overline{x_{2,2}^{(1)}} \cdot \overline{x_{1,2}^{(2)}} \cdot \overline{x_{2,1}^{(2)}} \cdot \overline{x_{1,1}^{(3)}} \cdot \overline{x_{2,2}^{(3)}}.$$

We can write similar expressions for all other y_j.

In this case, we have $I_0 = \{0,3,5,6\}$, $I_1 = \{1,2,4,7\}$ *and* $M_{11} = \{0\}$, $M_{12} = \{1\}$, $M_{21} = \{1\}$, $M_{22} = \{0\}$, *so*

$$z_{1,1} = z_{2,2} = y_0 \vee y_3 \vee y_5 \vee y_6,$$

$$z_{1,2} = z_{2,1} = y_1 \vee y_2 \vee y_4 \vee y_7.$$

Example 2. *Let* $n = 3, L = 2$ *(Figure 6). Then*

$$\pi_0 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \pi_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}, \pi_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix},$$

$$\pi_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \pi_4 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}, \pi_5 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

$$P_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, P_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$P_3 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, P_4 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, P_5 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

·	P_0	P_1	P_2	P_3	P_4	P_5
P_0	P_0	P_1	P_2	P_3	P_4	P_5
P_1	P_1	P_0	P_3	P_2	P_5	P_4
P_2	P_2	P_4	P_0	P_5	P_1	P_3
P_3	P_3	P_5	P_1	P_4	P_0	P_2
P_4	P_4	P_2	P_5	P_0	P_3	P_1
P_5	P_5	P_3	P_4	P_1	P_2	P_0

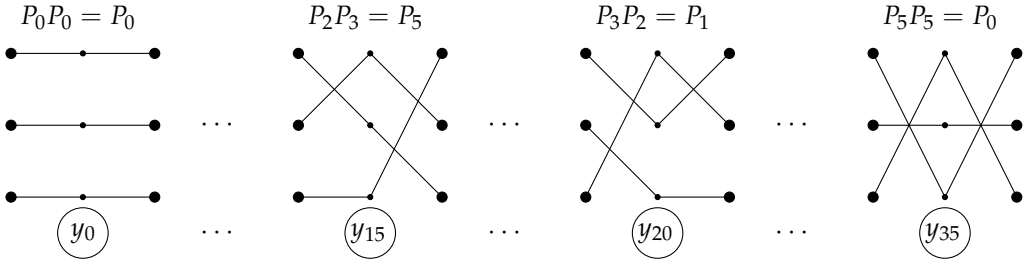

Figure 6. The case $n = 3, L = 2$.

$$X_1 = \begin{pmatrix} x_{11}^{(1)} & x_{12}^{(1)} & x_{13}^{(1)} \\ x_{21}^{(1)} & x_{22}^{(1)} & x_{23}^{(1)} \\ x_{31}^{(1)} & x_{32}^{(1)} & x_{33}^{(1)} \end{pmatrix}, X_2 = \begin{pmatrix} x_{11}^{(2)} & x_{12}^{(2)} & x_{13}^{(2)} \\ x_{21}^{(2)} & x_{22}^{(2)} & x_{23}^{(2)} \\ x_{31}^{(2)} & x_{32}^{(2)} & x_{33}^{(2)} \end{pmatrix}.$$

For example, we have $15 = 2 \cdot (3!) + 3$ *for* $j = 15$, *so*

$$y_{15} = x_{1,\pi_2(1)}^{(1)} \cdot x_{2,\pi_2(2)}^{(1)} \cdot x_{2,\pi_2(3)}^{(1)} \cdot x_{1,\pi_3(1)}^{(2)} \cdot x_{2,\pi_3(2)}^{(2)} \cdot x_{2,\pi_3(3)}^{(2)} \cdot \prod_{(\alpha,\beta)\neq(i,\pi_{a_{j,s}}(i))} \overline{x_{\alpha,\beta}^{(\gamma)}} =$$

$$x_{1,2}^{(1)} \cdot x_{2,1}^{(1)} \cdot x_{3,3}^{(1)} \cdot x_{1,2}^{(2)} \cdot x_{2,3}^{(2)} \cdot x_{3,1}^{(2)} \cdot \overline{x_{1,1}^{(1)}} \cdot \overline{x_{1,3}^{(1)}} \cdot \overline{x_{2,2}^{(1)}} \cdot \overline{x_{2,3}^{(1)}} \cdot \overline{x_{3,1}^{(1)}} \cdot \overline{x_{3,2}^{(1)}} \cdot$$
$$\overline{x_{1,1}^{(2)}} \cdot \overline{x_{1,3}^{(2)}} \cdot \overline{x_{2,1}^{(2)}} \cdot \overline{x_{2,2}^{(2)}} \cdot \overline{x_{3,2}^{(2)}} \cdot \overline{x_{3,3}^{(2)}}.$$

We can write similar expressions for the other y_j.
In this case, we have

$$M_{11} = \{0,1\}, M_{12} = \{2,3\}, M_{13} = \{4,5\},$$

$$M_{21} = \{2,4\}, M_{22} = \{0,5\}, M_{23} = \{1,3\},$$

$$M_{31} = \{3,5\}, M_{32} = \{1,4\}, M_{33} = \{0,2\}$$

and

$$I_0 = \{0,7,14,22,27,35\}, I_1 = \{1,6,16,20,29,33\}, I_2 = \{2,9,12,23,25,34\},$$

$$I_3 = \{3,8,17,18,28,31\}, I_4 = \{4,11,13,21,24,32\}, I_5 = \{5,10,15,19,26,30\}.$$

Thus

$$z_{1,1} = \bigvee_{s\in I_0\cup I_1} y_s = y_0 \vee y_7 \vee y_{14} \vee y_{22} \vee y_{27} \vee y_{35} \vee y_1 \vee y_6 \vee y_{16} \vee y_{20} \vee y_{29} \vee y_{33},$$

$$z_{1,2} = \bigvee_{s\in I_2\cup I_3} y_s = y_2 \vee y_9 \vee y_{12} \vee y_{23} \vee y_{25} \vee y_{34} \vee y_3 \vee y_8 \vee y_{17} \vee y_{18} \vee y_{28} \vee y_{31},$$

$$z_{1,3} = \bigvee_{s\in I_4\cup I_5} y_s = y_4 \vee y_{11} \vee y_{13} \vee y_{21} \vee y_{24} \vee y_{32} \vee y_5 \vee y_{10} \vee y_{15} \vee y_{19} \vee y_{26} \vee y_{30},$$

$$z_{2,1} = \bigvee_{s\in I_2\cup I_4} y_s = y_2 \vee y_9 \vee y_{12} \vee y_{23} \vee y_{25} \vee y_{34} \vee y_4 \vee y_{11} \vee y_{13} \vee y_{21} \vee y_{24} \vee y_{32},$$

$$z_{2,2} = \bigvee_{s \in I_0 \cup I_5} y_s = y_0 \vee y_7 \vee y_{14} \vee y_{22} \vee y_{27} \vee y_{35} \vee y_5 \vee y_{10} \vee y_{15} \vee y_{19} \vee y_{26} \vee y_{30},$$

$$z_{2,3} = \bigvee_{s \in I_1 \cup I_3} y_s = y_1 \vee y_6 \vee y_{16} \vee y_{20} \vee y_{29} \vee y_{33} \vee y_3 \vee y_8 \vee y_{17} \vee y_{18} \vee y_{28} \vee y_{31},$$

$$z_{3,1} = \bigvee_{s \in I_3 \cup I_5} y_s = y_3 \vee y_8 \vee y_{17} \vee y_{18} \vee y_{28} \vee y_{31} \vee y_5 \vee y_{10} \vee y_{15} \vee y_{19} \vee y_{26} \vee y_{30},$$

$$z_{3,2} = \bigvee_{s \in I_1 \cup I_4} y_s = y_1 \vee y_6 \vee y_{16} \vee y_{20} \vee y_{29} \vee y_{33} \vee y_4 \vee y_{11} \vee y_{13} \vee y_{21} \vee y_{24} \vee y_{32},$$

$$z_{3,3} = \bigvee_{s \in I_0 \cup I_2} y_s = y_0 \vee y_7 \vee y_{14} \vee y_{22} \vee y_{27} \vee y_{35} \vee y_2 \vee y_9 \vee y_{12} \vee y_{23} \vee y_{25} \vee y_{34}.$$

4. Deep Neural Network Solution

The calculation of the matrix $Z = X_1 \cdot X_2 \cdot \ldots \cdot X_L$ can be performed using a deep learning network, multiplying sequentially: $Y_1 = X_1$, $Y_k = Y_{k-1} \cdot X_k$, $(k = 2, \ldots, L)$. Then, $Z = Y_L$. The network diagram is shown in the Figure 7.

Let $X_k = (x_{ij}^{(k)})$, $Y_k = (y_{ij}^{(k)})$. Then,

$$y_{ij}^{(k)} = y_{i1}^{(k-1)} \cdot x_{1j}^{(k)} \oplus y_{i2}^{(k-1)} \cdot x_{2j}^{(k)} \oplus \ldots \oplus y_{in}^{(k-1)} \cdot x_{nj}^{(k)}.$$

To calculate the entries of matrices, we use the conjunction and addition modulo 2.

Theorem 3. *For travel maze problems $F(n, L)$, there is a deep neural network that has a depth of L,*

$$(2L-1)n^2$$

neurons and

$$2(L-1)n^3$$

connections between them.

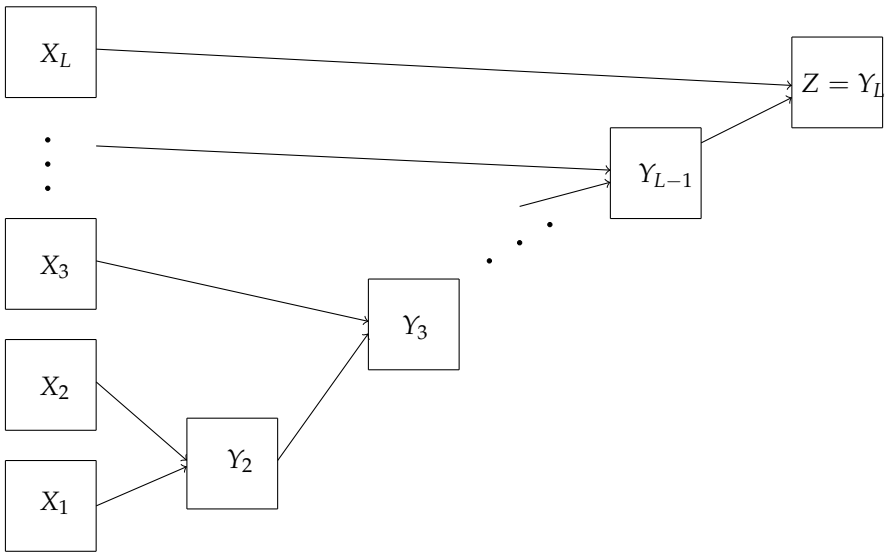

Figure 7. A deep neural network diagram for simplified travel maze problem.

5. Neural Network for *r*-Bounded Problem

The travel maze problem is called r-bounded $(0 \le r \le n-1)$ if the inequality $|\pi_{j_k}(i) - i| \le r$ holds for all $i = 1, \ldots, n; k = 1, \ldots, L$. This means that the corresponding permutation matrices are banded matrices with the bandwidth $r+1$, i.e., $x_{ij} = 0$ for $|i-j| \ge r+1$.

Let A be a banded matrix of the bandwidth $r+1$. The maximum number of nonzero entries in an arbitrary row of A is at most $2r+1$. The number of nonzero entries in A is at most

$$N_r = n^2 - (n-r)(n-r-1) = n(2r+1) - (r^2+r).$$

If A and B are banded matrices of the bandwidth $r+1$ and $t+1$, respectively, then the product AB is a banded matrix of bandwidth $r+t+1$.

Theorem 4. *For r-bounded travel maze problems $F(n, L)$, there is a shallow neural network that has a depth of 3,*

$$L \cdot N_r + n^2 + (n!)^L$$

neurons and

$$(L \cdot N_r + n^2) \cdot (n!)^L$$

connections between them.

Theorem 5. *For r-bounded travel maze problems $F(n, L)$, there is a deep neural network that has a depth of L,*

$$L \cdot N_r + \sum_{i=2}^{L} N_{ir} \text{ neurons if } Lr \le n-1,$$

$$L \cdot N_r + \sum_{i=2}^{[\frac{n-1}{r}]} N_{ir} + n^2 \cdot \left(L - \left[\frac{n-1}{r} \right] \right) \text{ neurons if } Lr > n-1,$$

$$2 \cdot \sum_{i=2}^{L} (ir+1) N_{ir} \text{ connections between neurons if } Lr \le n-1,$$

$$\text{and } 2 \cdot \sum_{i=2}^{[\frac{n-1}{r}]} (ir+1) N_{ir} + n^2 \cdot (2r+n+1) \cdot \left(L - \left[\frac{n-1}{r} \right] \right) \text{ connections if } Lr > n-1.$$

6. Conclusions and Outlook

- A shallow neural network combined from elementary Rosenblatt's perceptrons can solve the travel maze problem in accordance with Rosenblatt's first theorem.
- The complexity of the constructed solution of the travel maze problem for a deep network is much smaller than for the solution provided by the shallow network (the main terms are $2Ln^2$ versus $(n!)^L$ for the numbers of neurons and $2L^3$ versus $Ln^2(n!)^L$ for the numbers of connections).

The first result is important in the context of the widespread myth that elementary Rosenblatt's perceptrons have limited abilities and that Minsky and Papert revealed these limitations. This mythology has penetrated even into the encyclopedic literature [8].

Original Rosenblatt's perceptrons [1] (Figure 1) can solve any problem regarding the classification of binary images and, after minor modification, even wider. This simple fact was proven in Rosenblatt's first theorem, and nobody criticised this theorem and proof. The universal representation property of shallow neural networks were studied in the 1990s from different points of view, including the approximation of real-valued functions [9] and the evaluation of upper bounds on rates of approximation [10]. Elegant analysis of shallow neural networks involved infinite-dimensional hidden layers [11], and upper bounds were derived on the speed of decrease of approximation error as the number of network units increases. Abilities and limitations of shallow networks were reviewed recently in detail [12].

Of course, a single R-element can solve only linearly separable problems, and, obviously, not all problems are linearly separable. Stating this trivial statement does not require any intellectual effort. Minsky and Papert [5] considered much more complex systems then a single linear threshold R-element. They studied the same elementary perceptrons

that Rosenblatt did (Figure 1) with one restriction: *receptive fields of A-elements are bounded*. These limitations may assume a sufficiently small diameter of the receptive field (the most common condition) or a limited number of input connections of each A-neuron. Elementary perceptrons with such restrictions have limited abilities: if we have only local information, then we cannot solve such a global problem as checking the connectivity of a set or the travel maze problem with one glance. We should integrate the local knowledge into global criterion using a sequence of steps. This intuitively clear statement was accurately formalised and proved for the parity problem by Minsky and Papert [5].

Without restrictions, elementary perceptrons are omnipotent. In particular, they can solve the travel maze problem in the proposed form, but the complexity of solutions can be huge (Theorem 2). On the contrary, the deep network solution (Theorem 3) is much simpler and seems to be much more natural. It combines solutions from the one-step permutations locally, step by step, whereas the shallow network operates by all possible global paths. Restriction of the possible paths of travel by a bounded radius of a single step (Section 5) does not change the situation qualitatively. (The restricted problem is simpler than the original one. This should not be confused with the A-elements' receptive field limitations proposed by Minsky and Papert [5] that complicate all problems.)

The second observation seems to be more important than the first one: the properly selected deep solutions can be much simpler than the shallow solutions. In contrast to the widely discussed huge deep structures and their surprising efficiency (see the detailed exposition of the mathematics of deep learning in [13]) the relatively small but deep neural networks are non-surprisingly effective for the solution of problems where local information should be integrated into global decisions, such as in the discussed version of the travel maze problem. These networks combine the benefits of the fine-grained parallel procession and the solutions of problems at a glance with the possibility of emulating logic of sequential data analysis when it is necessary. The important question in this context is: "How deep should be the depth?" [14]. The answer depends on the problem.

Comparing a *lower bound* on the complexity of shallow networks with an *upper bound* on the complexity of deep networks for travel maze problem will provide a much deeper result than our theorems. We have not yet been able to obtain the lower complexity bound of shallow networks for travel maze problems. In this paper, our goal was more modest: we compared the Rosenblatt's first theorem "at work" with the obvious sequential deep algorithm of depth of order L. The solution of the simplified travel maze problem is equivalent to the computation of the product of L permutation matrices $X_1, X_2, \ldots, X_L$. The algorithm of depth L is equivalent to the parenthesization $((\ldots((X_1 \cdot X_2) \cdot X_3) \cdot \ldots) \cdot X_L$. Other parenthesizations can easily produce algorithms of lower depth of order $\log_2 L$. For example, for eight matrices, we can group multiplications as follows: $((X_1 \cdot X_2) \cdot (X_3 \cdot X_4)) \cdot ((X_5 \cdot X_6) \cdot (X_7 \cdot X_8))$. It is obvious that the minimal depth achievable by parenthesization for products of L square matrices growths with L as $\log_2 L$. (The theory of optimal parenthesization of rectangular matrix chain products is well-developed; see, for example, [15] and references there.) The *problem* is in the optimal computation of permutation matrix chain products with the depth independent of L. Is there any algorithm better than memorising the answers and the recognition of the input chains without any real computation of products? In other words, is there an algorithm that provides significantly more efficient solutions to the travel maze problem than the proof of Rosenblatt's first theorem does?

Thus, the open question remains: are the complexity estimates sharp? How far are our solutions from the best ones? We do not expect that this problem has a simple solution because even for multiplication of $n \times n$ matrices, no final solution has yet been found, despite great efforts and significant progress (to the best of our knowledge, the latest improvement from $n^{2.37287}$ to $n^{2.37286}$ was achieved recently [16]).

Another open question might attract attention: analysing the original geometric travel maze problem (see Figure 2 instead of its more algebraic simplification presented in Figure 3). It includes many non-trivial tasks, for example, convenient discrete representa-

tion of the possible paths with bounded curvature, lengths and ends, and the constructive selection of ϵ-networks in the space of such paths for preparing the input weights of A-elements.

Analysis of more realistic cases with differential paths and a possibility of stepping back requires geometric and topological methods. Differential paths with bounded curvature k and lengths l form a compact set Q_{kl} in the space of continuous paths equipped by the standard C^0 metrics ρ. A paths with its ε-vicinity form a strip that we see in Figure 2a. Each strip has its smooth mean path. We assume that the strips do not intersect beyond small neighborhoods of several isolated points of mean paths intersections. The ε-network in the set Q_{kl} and metrics ρ is finite. For an empirically given path, we have to find the closest paths from this ε-network. The additional challenge is in the discretization of the input information. We should present the paths as the broken lines on a grid. The selection of the grid should allow us to resolve the intersections. For reliable recognition, an assumption of the minimal possible angle α of mean path intersections or self-intersections is needed. After that, one can use the logic of Rosenblatt's first theorem and produce an elementary perceptron for solving the travel maze problem in this continuous setting. The perceptron construction and weights will depend on k, l, ε, and α. The detailed solution is beyond the scope of our paper.

The travel maze problem is discussed as a prototype problem for mobile robot navigation [17,18]. There exist universal "Elastic Principal Graph" (ElPiGraph) algorithms and open access software for the analysis of the raw image [7]. These algorithms were applied to the very new areas of data mining: they were used for revealing disease trajectories and dynamic phenotyping [19] and for the analysis of single-cell data for outlining the cell development trajectories [20] The travel maze problem in its differential settings (Figure 2a) was used as a benchmark for the ElPiGraph algorithm that untangles the complex paths of the maze [7].

The complexities of functions computable by deep and shallow networks used for the solution of the classification problem were compared for the same complexity of networks [21]. The complexities of the functions were measured using topological invariants of the superlevel sets. The results seem to support the idea that deep networks can address more difficult problems with the same number of resources.

The problem of effective parallelism pretends to be the central problem that is being solved by the whole of neuroinformatics [22]. It has long been known that the efficiency of parallel computations increases slower than the number of processors. There is a well known "Minsky hypothesis": the efficiency of a parallel system increases (approximately) proportionally to the logarithm of the number of processors; at the least, it is a concave function. Shallow neural networks pretend to solve all problems in one step, but the cost for that may be an enormous number of resources. Deep networks make possible a trade-off between resources (number of elements) and the time needed to solve a problem since they can combine the efficient parallelism of neural networks with elements of sequential reasoning. Therefore, neural networks can be a useful tool for solving the problem of efficient fine-grained parallel computing if we can answer the question: how deep should the depths be for different classes of problems? The case study presented in our note gives an example of a significant increase in efficiency for a reasonable choice of depth.

Author Contributions: Conceptualization, A.K.; methodology, A.K. and S.S.; formal analysis, A.K. and N.Z.; writing, A.K., S.S. and N.Z.; supervision, N.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Higher Education of the Russian Federation (agreement number 075-15-2020-808).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We are grateful to A.N. Gorban for his inspiring lectures about deep and shallow networks.

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

References

1. Rosenblatt, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*; Spartan Books: Washington, DC, USA, 1962.
2. Venkatesh, B.; Anuradha, J. A review of feature selection and its methods. *Cybern. Inf. Technol.* **2019,** *19*, 3–26. [CrossRef]
3. Al-Tashi, Q.; Abdulkadir, S. J.; Rais, H.M.; Mirjalili, S.; Alhussian, H. Approaches to multi-objective feature selection: A systematic literature review. *IEEE Access* **2020**, *8*, 125076–125096. [CrossRef]
4. Rong, M.; Gong, D.; Gao, X. Feature selection and its use in big data: Challenges, methods, and trends. *IEEE Access* **2019**, *7*, 19709–19725. [CrossRef]
5. Minsky, M.; Papert, S. *Perceptrons*; MIT Press: Cambridge, MA, USA,1988.
6. Seifert B.G. Binary classifiers, pcrceptrons and connectedness in metric spaces and graphs. *J. Pure Appl. Algebra* **1992**, *83*, 197–203. [CrossRef]
7. Albergante L.; Mirkes E.; Bac J.; Chen H.; Martin A.; Faure L.; Barillot E.; Pinello L.; Gorban A.; Zinovyev A. Robust and Scalable Learning of Complex Intrinsic Dataset Geometry via ElPiGraph. *Entropy* **2020**, *22*, 296. [CrossRef] [PubMed]
8. Bishop, J.M. History and Philosophy of Neural Networks. In *Encyclopedia of Life Support Systems (EOLSS): Computational Intelligence*; Ishibuchi, H., Ed.; UNESCO: Eolss Publishers, Paris, France, 2015.
9. Ito, Y. Finite mapping by neural networks and truth functions. *Math. Sci.* **1992**, *17*, 69–77.
10. Kůrková, V.; Savický, P.; Hlavávcková, K. Representations and rates of approximation of real-valued Boolean functions by neural networks. *Neural Netw.* **1998**, *11*, 651–659. [CrossRef]
11. Kainen, P.C.; Kůrková, V. An integral upper bound for neural network approximation. *Neural Comput.* **2009**, *21*, 2970–2989. [CrossRef] [PubMed]
12. Kůrková, V. Limitations of Shallow Networks. In *Recent Trends in Learning From Data. Studies in Computational Intelligence*; Oneto, L., Navarin, N., Sperduti, A., Anguita, D., Eds.; Springer: Cham, Switzerland, 2020; Volume 896, pp. 129–154._6 [CrossRef]
13. Berner, J.; Grohs, P.; Kutyniok, G.; Petersen P. The Modern Mathematics of Deep Learning. *arXiv* **2021**, arXiv:2105.04026.
14. Gorban, A.N.; Mirkes, E. M.; Tyukin, I.Y. How deep should be the depth of convolutional neural networks: A backyard dog case study. *Cogn. Comput.* **2020**, *12*, 388–397. [CrossRef]
15. Weiss, E.; Schwartz, O. Computation of Matrix Chain Products on Parallel Machines. In Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil, 20–24 May 2019; pp. 491–500. [CrossRef]
16. Alman, J.; Vassilevska Williams, V. A Refined Laser Method and Faster Matrix Multiplication. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), Virtual, 10–13 January 2021; pp. 522–539. [CrossRef]
17. Gupta, B.; Sehgal, S. Survey on techniques used in autonomous maze solving robot. In Proceedings of the 2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence), Noida, India, 25–26 September 2014; pp. 323–328. [CrossRef]
18. Wu, C.M.; Liaw, D.C.; Lee, H.T. A method for finding the routes of mazes. In Proceedings of the 2018 International Automatic Control Conference (CACS), Taoyuan, Taiwan, 4–7 November 2018; pp. 1–4. [CrossRef]
19. Golovenkin, S.E.; Bac, J.; Chervov, A.; Mirkes, E.M.; Orlova, Y.V.; Barillot, E.; Gorban, A.N.; Zinovyev, A. Trajectories, bifurcations, and pseudo-time in large clinical datasets: Applications to myocardial infarction and diabetes data. *GigaScience* **2020**, *9*, giaa128. [CrossRef] [PubMed]
20. Chen, H.; Albergante, L.; Hsu, J.Y.; Lareau, C.A.; Lo Bosco, G.; Guan, J.; Zhou, S.; Gorban, A.N.; Bauer, D.E.; Aryee, M.J.; et al. Single-cell trajectories reconstruction, exploration and mapping of omics data with STREAM. *Nat. Commun.* **2019**, *10*, 1–14. [CrossRef] [PubMed]
21. Bianchini, M.; Scarselli, F. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 1553–1565. [CrossRef] [PubMed]
22. Gorban, A.N. Neuroinformatics: What are us, where are we going, how to measure our way? A lecture given at the USA-NIS Neurocomputing opportunities workshop, Washington DC, July 1999 (Associated with IJCNN'99). *arXiv* **2003**, arXiv:cond-mat/0307346. [CrossRef]