Journal of Membrane Computing (2024) 6:29–41
https://doi.org/10.1007/s41965-024-00136-1

RESEARCH PAPER

Implementing perceptrons by means of water-based computing

Nicoló Civiero[1] · Alec Henderson[2] · Thomas Hinze[3] · Radu Nicolescu[4] · Claudio Zandron[1]

Received: 12 December 2023 / Accepted: 2 February 2024 / Published online: 27 February 2024

Abstract
Water-based computing emerged as a branch of membrane computing in which water tanks act as permeable membranes connected via pipes. Valves residing at the pipes control the flow of water in terms of processing rules. Resulting water tank systems provide a promising platform for exploration and for case studies of information processing by flow of liquid media like water. We first discuss the possibility of realizing a single layer neural network using tanks and pipes systems. Moreover, we discuss the possibility to create a multi-layer neural network, which could be used to solve more complex problems. Two different implementations are considered: in a first solution, the weight values of the connections between the network nodes are represented by tanks. This means that the network diagram includes multiplication structures between the weight tanks and the input tanks. The second solution aims at simplifying the network proposed in the previous implementation, by considering the possibility to modify the weight values associated to neuron by varying the diameter of the connecting pipes between the tanks. The multiplication structures are replaced with a timer that regulates the opening of the outlet valves of all the tanks. These two implementations can be compared to evaluate their efficiency, and considerations will be made regarding the simplicity of implementation.

Keywords Membrane systems · Water-based computing · Neural networks

1 Introduction

P systems, initially introduced by Gh. Păun in [27], are a computational model inspired by biological membranes that operate in a parallel and distributed manner. These systems are characterized by their decentralized nature and their evolution is based on the content of interconnected membranes. A strong investigation effort has been done on the model, and it is still in progress, considering different aspects. Recent works appeared considering questions related to computing properties [22, 25], computing efficiency [1, 16, 17], relations with other formal models like, e.g., Petri nets [4], Morphogenetic systems [34], or Markov chains [32], and application to real problems [3, 7, 30, 35, 38].

Numerous variants of P systems have also been proposed and extensively studied, including P systems with active membranes [26, 28, 33], spiking neural P systems [6, 9, 13, 19, 31, 37, 40], tissue P systems [15, 23, 38], and P colonies [5, 14].

Recent research efforts have focused on simulating P systems on mainstream hardware [2, 36], formal verification techniques [20, 21], or employing more visual approaches like [8].

Another recently introduced idea concerns membrane water computing, introduced in [10] and driven by the goal of obtaining a parallel computing system without any central control. In such a system, the flow of water is solely regulated by local measurements of tank filling levels in a

✉ Claudio Zandron
claudio.zandron@unimib.it

Alec Henderson
alec.henderson@jcu.edu.au

Thomas Hinze
thomas.hinze@uni-jena.de

Radu Nicolescu
r.nicolescu@auckland.ac.nz

[1] Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo), Università degli Studi di Milano-Bicocca, Viale Sarca 336, 20126 Milan, Italy

[2] Australian Institute of Tropical Health and Medicine, James Cook University, Townsville, Australia

[3] Department of Bioinformatics, Friedrich Schiller University Jena, Ernst-Abbe-Platz 2, 07743 Jena, Germany

[4] University of Auckland, School of Computer Science, Auckland, New Zealand

finite number of water tanks, each capable of holding an initial volume of water and storing or collecting water up to a maximum capacity. The water tank system can be viewed as a membrane system: the water tanks can be seen as membranes permeable to inflow and/or outflow of water molecules, whose presence is dynamically regulated by local measurements (interaction rules).

The volume of water contained in a tank serves as both data carrier and a medium for data processing, achieved by manipulating the volume over time. Tanks are interconnected using pipes, which allow the directed transfer of water from one tank to another when opened. The pipes can be equipped with one or more valves, which can be configured in different ways. A valve has two states: "fully open" or "fully closed", and it is determined by monitoring the filling level in a specific tank. When the level of water exceeds a predetermined threshold or indicates a nearly empty tank, the valve fully opens or remains closed at the hosting pipe during the ongoing time step, respectively. A pipe will transport water only if all its valves are fully opened and the supplying tank has water available. The entrance of a pipe can be positioned at any desired filling level in its supply tank, requiring a minimum amount of water in the tank before the pipe can be filled.

Water tank systems provide a promising platform for exploration and for case studies of information processing based on the controlled flow of liquid media like water. This concept gives a strong motivation and substantiates the significance of further work in detail for application scenarios.

A water tank system can operate in either analog or binary mode. In analog mode, the volume of water within a tank represents a non-zero natural number. To facilitate this, we introduce water tank systems for arithmetic operations such as addition, non-negative subtraction, division, and multiplication. These systems can be assembled to perform sequenced or nested computations. Furthermore, a ring oscillator, consisting of a cyclic structure with at least three water tanks, emulates a clock signal. In binary mode, an empty or nearly empty water tank corresponds to the logical value “0,” while a full or nearly full tank corresponds to “1,” with latencies during the filling or emptying process.

The obtained systems operate autonomously in a decentralized manner, simply relying on local measurements of filling levels. We stress the fact that, since a water tank can be viewed as a membrane that allows the inflow and/or outflow of water molecules, dynamically regulated by local measurements, such an approach is closely related to tissue membrane systems.

In the original paper [10], authors define basic logic gates such as OR, AND, and a bit duplicator for water-based logic operations. These logic gates can be connected to form Boolean circuits with the ability of inherent self-synchronization, eliminating the need for external control.

In this paper, we first discuss the possibility of realizing a single layer neural network using tanks and pipes systems, through which water flows. Moreover, we discuss the possibility to create a multi-layer neural network, which could be used to solve more complex problems. We stress the fact that one advantage of such an implementation lies in the possibility to adopt it for explaining the functioning of Neural Network at different levels, to students or even more general audience, clearly illustrating the basic principles behind a Neural Network. In fact, the flow and the containment of water are easily visible by human senses, and the process of learning can be directly observed in details.

Two different implementations are considered: in a first solution, the weight values of the connections between the network nodes are obtained by using specific tanks. The second solution aims at simplifying the network proposed in the previous one, by considering the possibility to modify the weight values associated to each neuron by varying the diameter of the connecting pipes between the tanks.

The paper is organized as follows. In Sect. 2, we recall some definitions related to water based computing, and we recall some basic multiplication schemes realized by means of water tanks, which will be used in the rest of the paper. In Sect. 3, two different implementations of the basic perceptron are presented. In Sect. 4, an implementation of the multilayer-perceptron and the description of three different activation functions are discussed. In Sect. 5, we draw some conclusions and give some directions for future investigations.

2 Basic definitions

In this section, we shortly recall some definitions that will be useful while reading the rest of the paper. For a complete introduction to P systems, we refer the reader to *The Oxford Handbook of Membrane Computing* [29].

A *water tank system* represents a special type of membrane systems in which a single membrane is described by a *water tank* able to store an amount of water up to its predefined finite capacity. The communication between membranes has been managed by *pipes* that enable a controllable flow of water from one tank to another one. Communication rules appear by definition of *valves*. Here, each pipe can be equipped with an arbitrary number of valves. By default, a valve fully closes its hosting pipe. A valve either fully opens or remains closed its hosting pipe based on measurements iterated in discrete time steps. For instance, a valve opens if and only if the filling level in a specific water tank exceeds a certain threshold, otherwise it closes. If the condition for an open valve is not fulfilled any more, it closes at the end of the ongoing time step.

Water gets transported via a pipe if and only if all residing valves are opened and the supply tank contains water.

The first formal definition of a water tank system was given in [10]. Later, a more simplified version was published in [11, 12]. In order to cope with the needs for emulation of perceptrons, the modelling framework for water tank systems undergoes a further stage of extension by additional types of valves and by additional parameters for specification of pipes.

Formally, a water tank system is a construct

$$\Pi = (W, A, \tau, E, r, P, v_0, s_0, \Delta t) \tag{1}$$

with its components:

- W is a finite and non-empty set of tank identifiers (water tanks).
- A is a finite and non-empty set of valve identifiers (actuators).
- $\tau : W \longrightarrow \mathbb{R}_+ \cup \{\infty\}$ is a function assigning a capacity to each tank (tank capacity). $\mathbb{R}_+$ stands for the set of positive rational numbers. The capacity defines the maximum volume of water a tank can store. Excessive water is removed from a tank by overflow drain. Please note that tanks with an infinite capacity are allowed to act as a reservoir.
- $E \subset W \times \{<, =, \leq, >, \geq, \neq\} \times \mathbb{R}$ specifies a finite set of decision rules resulting from measurements (evaluation). A measurement reveals the current volume of water in a tank from W. We assume that each measurement returns a non-negative rational number in $\mathbb{R}$ including zero. An element $e \in E$ stands for a comparison by means of a relational operator. This comparison is carried out what finally implies an underlying decision by answering "true" or "false", respectively.
- $r : A \longrightarrow E$ defines a mapping that assigns a decision rule to each valve. Hence, each valve comes with a dedicated behaviour (reaction).
- $P \subset W \times W \times \mathcal{P}(A)$ symbolises a finite set of pipes in which each pipe starts at a tank from W, ends at a tank from W and hosts none, one, or several valves. These valves have been given by an element from the power set $\mathcal{P}(A)$.
- $v_0 : W \longrightarrow \mathbb{R} \cup \{\infty\}$ specifies the initial volume of water for each water tank in W. For all water tanks $w \in W$, it holds $v_0(w) \leq \tau(w)$.
- $s_0 : P \longrightarrow \mathbb{R}_+$. This function assigns an initial diameter (size) to each pipe. Diameters are expressed by rational numbers greater than zero.
- $\Delta t \in \mathbb{R}_+$ defines the duration of a discrete time step given by a constant non-negative rational number.

A water tank system evolves in discrete constant time steps beginning with its initial configuration. The execution of a time step consists of a sequence of actions. Valves are closed by default. First, all measurements are done simultaneously in all involved tanks. Then, all decisions based on these measurements have been made. Next, the valves update their state according to the corresponding decision rules. In case, a decision ends up with "true", the valve fully opens. Otherwise, the valve remains closed. Now, water can flow or not through the pipes. Each pipe whose valves are all fully opened transports a portion of water during the ongoing time step when supplied. In addition, the portion of water depends on the size of the pipe. As a consequence, the water volume of either related tanks needs to be updated (increased or decreased). Finally, the size of each pipe can be adapted (made smaller or larger). After all the aforementioned actions have been carried out, the processing within the current time step is finalised, all valves become closed again, and the subsequent time step might begin. The water tank system stops if the water volumes in all water tanks keep constant over two successive time steps indicating a final system's configuration.

We recall now the integer multiplication scheme, a copy of the one presented in [10]. In this scheme, the operation works as follows: a unit is subtracted from tank x at each iteration of the loop until the tank x becomes empty, while at each iteration, the value of y is added to the result tank. For implementation details and a detailed description of how it works, we refer the reader to [10] (Fig. 1).

For the sake of completeness, the schemes related to multiplication with rational values are also provided. However, it is important to keep in mind that multiplication involving rational numbers introduces approximations and lacks precision. For optimal outcomes, we have categorized the multiplication process into three cases, taking into account the input values (x, y).

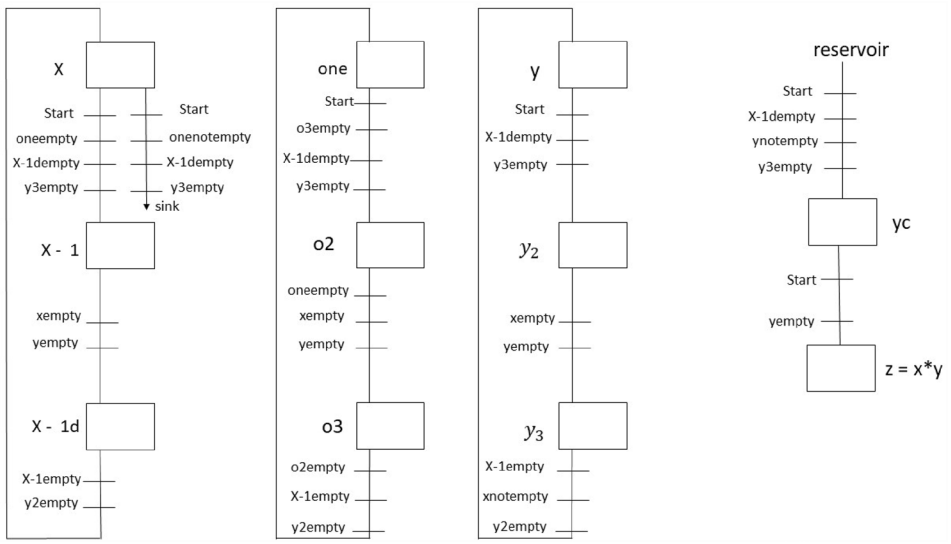
1. $x, y < 0.81$
2. $x, y < 1$ and x or $y > 0.8$
3. x or $y > 1$

We depict below the schemes for case 1 (see Fig. 2) and case 2 (see Fig. 3); the subtraction schema can be found in [10].

The subtraction works as follows: the valves placed on the pipes connecting the tanks (x and y) to the sink are opened simultaneously. The contents of both tanks are drained at the same time until one of the tanks becomes empty. The result is then taken from tank x. From this description, it is evident that if x is less than y, then the result of the operation will be zero.

For the third case, the schema is the same as the multiplication with integers presented earlier. However, it is

Fig. 1 Integer multiplication



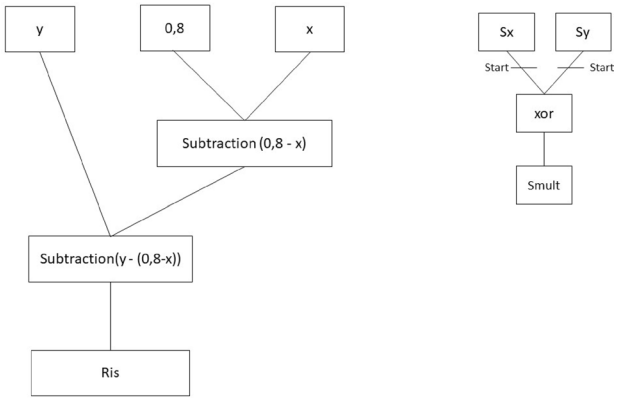


Fig. 2 Rational values multiplication case 1

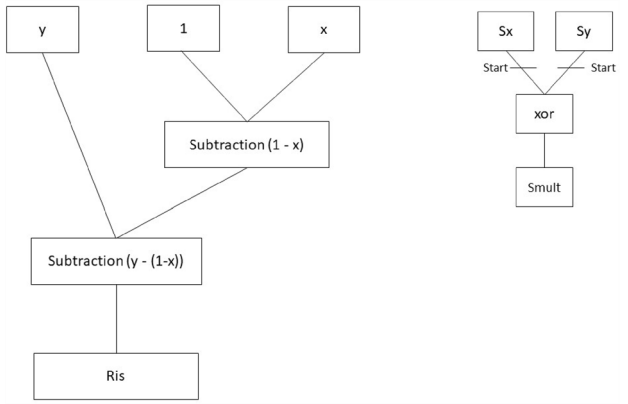


Fig. 3 Rational values multiplication case 2

important to note that, for this operation, it is advisable to position the larger number in the tank labeled as x, to ensure a more accurate result. Conversely, if the numbers are swapped, the outcome may not be as precise; as an example, the operation 0.3 * 5.5 would give a result of 5.5.

3 Implementing perceptrons through water-based computing

Artificial Neural Networks (or simply Neural Networks) are mathematical models composed of nodes (neurons) that are inspired by the functioning of the human brain, where interconnected neurons exchange information. A neural network is an "adaptive" system capable of modifying its structure (nodes, interconnections, and weights) based on both external data and internal information that connect and pass through the neural network during the learning phase.

The perceptron, introduced by McCulloch-Pitts in [24], is a machine learning algorithm used for supervised learning of binary classifiers. These classifiers are functions that determine whether an input, represented by a numerical vector, belongs to a particular class or not. More formally, given an input with n variables $(x_1, x_2, ..., x_k)$, the algorithm define a boundary as a linear combination of these variables: $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = 0$, where $w_i, 1 \leq i \leq n$, are the weights and b is a constant called *bias*. The perceptron algorithm determines values for $w_i, 1 \leq i \leq n$, and b in such a way that the data points on one side of the line belong to one class, while the data points on the other side belong to the other class. If $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b > 0$, then the classifier outputs 1; otherwise, it outputs 0. A learning algorithm to determine the weights consists in randomly assign a value to each of them, initially, and then iteratively updating the values according to some training dataset, until a convergence criterion has been reached.

In this section, we propose an implementation of a perceptron and a simple multilayer perceptron by means of

Water-Based Computing. In particular, we will consider water-based systems operating in analog mode: in this mode, the volume of water within a tank corresponds to a non-zero natural number. A XOR gate is used to track negative or positive results: we assume that positive values encode to logical value '0' and negative value to '1'. The XOR gate can be obtained as a modified version of the OR gate presented in [10], with an added valve in the result tank.

The water volume from the two input tanks is combined in a result tank. The input tanks have a maximum capacity corresponding to the logic level 1, while the result tank has two times this capacity. When both inputs tanks are 1, the result of the XOR gate must be 0. This issue can be resolved by using a simple valve added to the original OR gate, that opens when the content of the result tank reaches its full level. All the water is then drained, resulting in an empty result tank, corresponding to a 0. This logic gate will be used to control the sign of the operations of multiplication used to design perceptrons.

As a starting point, a simple version of a network with a single node of binary activation (0,1) was considered. This node is called a "simple perceptron", because it uses a simple step function as its activation function. The activation function of a node is a mathematical function that defines the output of the node after receiving the sum of weighted inputs.

We discuss two possible implementations of the network, referred to as implementation 1 and implementation 2. We stress the fact that the subtractions used in the schemes of the two presented solutions allow for negative results. The operation is similar to the non-negative subtraction schema presented in [10]. Water is simultaneously discharged from both tanks until one becomes empty. At this point, the remaining water in the other tank flows towards the result tank. Additionally, on the right side of the diagrams in figures 2 and 3, there is a control tank which indicates if the subtraction result is negative.

3.1 Implementation 1

In the first proposed implementation, the chosen approach is to multiply the input by the corresponding weight value. Formally, the system to implement the perceptron is defined as follows:

$W = \{(w1), (x1), (w2), (x2), (b), (S+),$
$(S-), (Res), (Sw1), (Sx1), (Sw2),$
$(Sx2), (Sb), (Smultiplication1),$
$(Smultiplication2), ([(S+) - (S-)]),$
$(w1old), (t), (y), (X1), (n),$
$(Subtraction1), (Ssub1), (SX1), (xor),$
$(SMult3), (sw1new), (sw1old),$
$(Subtraction2), (w1new), (s'), (g'),$
$(h'), (f'), (e'), (reservoir)\}$

$A = \{(w1newne), (w2newne), (bnewne),$
$(sw1newne), (sw2newne), (sbnewne),$
$(e'ne), (Start), (SMult1e),$
$(SMult1ne), (SMult2e), (SMult2ne),$
$(Sbe), (Sbne), (be), (w1ne), (Resne),$
$(x1ne), (sw1olde), (s'ne),$
$(f'ne), (g'ne), (h'ne), (sw1oldne),$
$(Smult3e), (Smult3ne), (te), (yne),$
$(sw1ne), (ye), (X1e), (Sub1e), (ne), (Mult3e),$
$(w1olde), (Mult4e), (SSubtraction2ne)\}$

$\tau = \{(w1, 2), (x1, 2), (w2, 2), (x2, 2),$
$(b, 2), (S+, 10), (S-, 10), (Res, 2),$
$(Sw1, 1), (Sx1, 1), (Sw2, 1), (Sx2, 1),$
$(Sb, 1), (Smultiplication1, 1),$
$(Smultiplication2, 1), (xor, 1),$
$([(S+) - (S-)], 10), (reservoir, \infty)$
$(w1old, 2), (t, 2), (y, 2), (X1, 2),$
$(n, 1), (Subtraction1, 2), (Ssub1, 1),$
$(SX1, 1), (SMult3, 1), (sw1new, 1),$
$(sw1old, 1), (Subtraction2, 2),$
$(w1new, 2), (s', 1), (g', 1), (h', 1), (f', 1), (e', 1)\}$

$E = \{(w1new > 0), (w2new > 0), (bnew > 0),$
$(sw1new > 0), (sw2new > 0),$
$(sbnew > 0), (Smultiplication1 > 0),$
$(Smultiplication2 > 0), (Sb > 0),$
$(b > 0), (w1 > 0), (Res > 0), (x1 > 0),$
$(s' > 0), (Sw1old > 0), (f' > 0),$
$(g' > 0), (h' > 0), (e' > 0), (t > 0),$
$(y > 0), (Sw1 > 0), (X1 > 0),$
$(Sub1 > 0), (n > 0), (Multiplication3 > 0), (w1old > 0),$
$(Multiplication4 > 0), (Ssubtraction2 > 0), (SMult3 > 0)\}$

r=

$$w1newne = \begin{cases} 1 \text{ if } \textit{Volume of } w1new > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$w2newne = \begin{cases} 1 \text{ if } \textit{Volume of } w2new > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$bnewne = \begin{cases} 1 \text{ if } \textit{Volume of } bnew > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$sw1newne = \begin{cases} 1 \text{ if } \textit{Volume of } sw1new > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$sw2newne = \begin{cases} 1 \text{ if } \textit{Volume of } sw2new > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$sbnewne = \begin{cases} 1 \text{ if } \textit{Volume of } sbnew > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$e'ne = \begin{cases} 1 \text{ if } \textit{Volume of } e' > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$start = \begin{cases} 1 \text{ if } \textit{time} > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$SMult1e = \begin{cases} 1 \text{ if } \textit{Volume of } Smultiplication1 = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$SMult1ne = \begin{cases} 1 \text{ if } \textit{Volume of } Smultiplication1 > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$Smult2e = \begin{cases} 1 \text{ if } \textit{Volume of } Smultiplication2 = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$Smult2ne = \begin{cases} 1 \text{ if } \textit{Volume of } Smultiplication2 > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$Sbe = \begin{cases} 1 \text{ if } \textit{Volume of } Sb = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$Sbne = \begin{cases} 1 \text{ if } \textit{Volume of } Sb > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$be = \begin{cases} 1 \text{ if } \textit{Volume of } b = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$w1ne = \begin{cases} 1 \text{ if } \textit{Volume of } w1 > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$Resne = \begin{cases} 1 \text{ if } \textit{Volume of } Res > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$x1ne = \begin{cases} 1 \text{ if } \textit{Volume of } x1 > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$sw1olde = \begin{cases} 1 \text{ if } \textit{Volume of } sw1old = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$sw1oldne = \begin{cases} 1 \text{ if } \textit{Volume of } sw1old > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$s'ne = \begin{cases} 1 \text{ if } \textit{Volume of } s' > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$g'ne = \begin{cases} 1 \text{ if } \textit{Volume of } g' > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$h'ne = \begin{cases} 1 \text{ if } \textit{Volume of } h' > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$e'ne = \begin{cases} 1 \text{ if } \textit{Volume of } e' > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$f'ne = \begin{cases} 1 \text{ if } \textit{Volume of } f' > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$SMult3e = \begin{cases} 1 \text{ if } \textit{Volume of } SMult3 = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$SMult3ne = \begin{cases} 1 \text{ if } \textit{Volume of } SMult3 > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$SSubtraction2ne = \begin{cases} 1 \text{ if } \textit{Volume of } SSubtraction2 > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$te = \begin{cases} 1 \text{ if } \textit{Volume of } t = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$yne = \begin{cases} 1 \text{ if } \textit{Volume of } y > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$ye = \begin{cases} 1 \text{ if } \textit{Volume of } y = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$sw1ne = \begin{cases} 1 \text{ if } \textit{Volume of } sw1 > 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$X1e = \begin{cases} 1 \text{ if } \textit{Volume of } X1 = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$Sub1e = \begin{cases} 1 \text{ if } \textit{Volume of } Subtraction1 = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$ne = \begin{cases} 1 \text{ if } n = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$Mult3e = \begin{cases} 1 \text{ if } \textit{Volume of } Multiplication3 = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$w1olde = \begin{cases} 1 \text{ if } \textit{Volume of } w1old = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

$$Mult4e = \begin{cases} 1 \text{ if } \textit{Volume of } Multiplication4 = 0 \\ 0 \text{ if } \textit{otherwise} \end{cases}$$

Meaning that 1 marks the valve to be open and 0 closed, respectively.

$P = \{(reservoir, w1, \{w1newne\}), (w1, Multiplication1, \{start\}),$
$(x1, Multiplication1, \{start\}),$
$(reservoir, w2, \{w2newne\}), (w2, Multiplication2,$
$\{start\}), (x2, Multiplication2, \{start\}),$
$(reservoir, b, \{bnewne\}), (Multiplication1, S+,$
$\{SMult1e\}), (Multiplication1, S-,$
$\{SMult1ne\}), (Multiplication2, S+, \{SMult2e\}),$
$(Multiplication2, S-, \{SMult2ne\}),$
$(b, S+, \{Sbe\}), (b, S-, \{Sbne\}),$
$(S+, [(S+) - (S-)], \{\}), (S-, [(S+) - (S-)], \{\}),$
$([(S+) - (S-)], Res, \{\}), (Res, sink, \{\}),$
$(reservoir, Sw1, \{sw1newne, e'ne\}),$
$(reservoir, Sw2, \{sw2newne, e'ne\}),$
$(reservoir, Sb, \{sbnewne, e'ne\}), (Sw1, xor, \{start\}),$
$(Sx1, xor, \{start\}), (Sw2, xor, \{start\}),$
$(Sx2, xor, \{start\}), Sb, sink, \{start, be\}),$
$(xor, Smultiplication1, \{\}), (xor, Smultiplication2, \{\}),$
$(reservoir, w1old, \{w1ne\}),$
$(reservoir, y, \{Resne\}), (reservoir, X1, \{x1ne\}),$
$(reservoir, Ssub1, \{start, te, yne\}),$
$(reservoir, sw1old, \{Sw1ne\}), (reservoir, s', \{start\}),$
$(reservoir, sw1new, \{SSubtraction2ne\}),$
$(w1old, S+, \{sw1olde, f'ne\}),$
$(w1old, S-, \{sw1oldne, f'ne\}), (t, Subtraction1, \{s'ne\}),$
$(y, Subtraction1, \{s'ne\}), (Subtraction1, Multiplication3, \{g'ne\}),$
$(X1, Multiplication3, \{g'ne\}),$
$(Multiplication3, Multiplication4, \{h'ne\}),$
$(n, Multiplication4, \{h'ne\}), (Multiplication4, S+, \{Smult3e\}),$
$(Multiplication4, S-, \{Smult3ne\}), (S+, Subtraction2, \{e'ne\}),$
$(S-, Subtraction2, \{e'ne\}), (Subtraction2, w1new, \{\}),$
$(w1new, Sink, \{e'ne\}), (Ssub1, xor, \{g'ne\}), (SX1, xor, \{g'ne\}),$
$(xor, Smult3, \{\}), (sw1new, Sink, \{e'ne\}),$
$(sw1old, Sink, \{e'ne\}), (s', g', \{te, ye\}),$
$(g', h', \{X1e, Sub1e\}), (h', f', \{ne, Mult3e\}),$
$(f', e', \{w1olde, Mult4e\}), (e' Sink, \{Start\})\}$

$v_0 = \{(w1;0), (x1;0), (w2;0), (x2;0),$
$(b;0), (S+;0), (S-;0), (Res;0), (Sw1;0),$
$(Sx1;0), (Sw2;0), (Sx2;0), (Sb;0),$
$(Smultiplication1;0), (Smultiplication2;0),$
$(xor;0), ([(S+) - (S-)];0), (w1old;0),$
$(t;0), (y;0), (X1;0),$
$(n;0), (Subtraction1;0), (Ssub1;0), (SX1;0),$
$(SMult3;0), (sw1old;0), (sw1new;0),$
$(Subtraction2;0), (w1new;0), (s';0),$
$(g';0), (h';0), (f';0), (e';0)\}$

We have decided to set a maximum size of 2 for the input and weight tanks to keep the network dimensions limited. For the control tanks, the maximum value is set to one because the possible values of a control tank are either 1 or 0.

To represent the weight of the connection between the input and the node, a tank is used, and the content of the tank corresponds to the weight value (see Fig. 4).

On the left side of the schema, the actual network is depicted, with tanks for the input (x) and tanks for the weights (W). In this hypothesis, these values are multiplied together (multiplication1, multiplication2) using the multiplication schema mentioned earlier. The result of the multiplication then flows into the tanks S (sum), while observing the value contained in the tanks Smultiplication1 and Smultiplication2 (multiplication sign). The tank (S+) contains positive values, while the tank (S-) contains negative values. The Bias value is added to one of these two tanks, depending on the value of the tank Sb (bias sign). The subtraction is then performed between the values contained in the tank (S+) and (S-) to obtain the result of the network, which is then passed through the activation function to obtain the network's output.

On the right side, there are control tanks used to calculate the signs of multiplication and bias. In particular, the XOR operation is used for the multiplication sign, and the values contained in the tanks Smultiplication (SMult3) will be 0 for a positive sign and 1 for a negative sign. The Feed-Forward phase is then followed by the weight update phase (see Fig. 5).

On the left side of the schema, the subtraction between the desired value (t) and the value obtained from the network (y) is performed first. Then, the multiplications between the result of the subtraction, the input, and the learning rate are carried out. On the right side of the schema, control tanks are present for the sign value of the multiplication (SMult3), sign value of old weight(Sw1old), and sign value of new weight(Sw1new). Finally, there is a column of control tanks used to adjust the valves of the input tanks to carry out the operations in the correct order.

3.2 Implementation 2

In the second proposed implementation, it has been chosen not to use tanks for the weights but instead variable-sized pipes that connect the tanks. Formally, the system to implement the perceptron is defined as follows:

$$W = \{(x1), (x2), (b), (S+), (S-), [(S+) - (S-)], (Res), (Sw1), (Sx1), (Sw2), (Sx2), (Sb), (Smultiplication1), (Smultiplication2), (xor), (reservoir)\}$$

$$A = \{(bnewne), (sw1newne), (sw2newne), (sbnewne), (e'ne), (Start), (Smult1e), (Smult1ne), (Smult2e), (Smult2ne), (Sbe), (Sbne), (be), (v)\}$$

$$\tau = \{(x1, 2), (x2, 2), (b, 2), (S+, 10), (S-, 10), ([(S+) - (S-)], 10)(Res, 2), (Sw1, 1), (Sx1, 1), (Sw2, 1), (Sx2, 1), (Sb, 1), (Smultiplication1, 1), (Smultiplication2, 1), (xor, 1), (reservoir, \infty)\}$$

$$E = \{(bnew > 0), (sw1new > 0), (sw2new > 0), (sbnew > 0), (e' > 0), (Smultiplication1 > 0), (Smultiplication2 > 0), (Sb > 0), (b > 0), (v < n)\}$$

r=

$$bnewne = \begin{cases} 1 \text{ if } Volume\ of\ bnew > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$sw1newne = \begin{cases} 1 \text{ if } Volume\ of\ sw1new > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$sw2newne = \begin{cases} 1 \text{ if } Volume\ of\ sw2new > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$sbnewne = \begin{cases} 1 \text{ if } Volume\ of\ sbnew > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$e'ne = \begin{cases} 1 \text{ if } Volume\ of\ e' > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$start = \begin{cases} 1 \text{ if } time > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$Smult1e = \begin{cases} 1 \text{ if } Volume\ of\ Smultiplication1 = 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$Smult1ne = \begin{cases} 1 \text{ if } Volume\ of\ Smultiplication1 > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$Smult2e = \begin{cases} 1 \text{ if } Volume\ of\ Smultiplication2 = 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$Smult2ne = \begin{cases} 1 \text{ if } Volume\ of\ Smultiplication2 > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$Sbe = \begin{cases} 1 \text{ if } Volume\ of\ Sb = 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$Sbne = \begin{cases} 1 \text{ if } Volume\ of\ Sb > 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$be = \begin{cases} 1 \text{ if } Volume\ of\ b = 0 \\ 0 \text{ if } otherwise \end{cases}$$

$$v = \begin{cases} 1 \text{ if } Volume\ of\ timer\ tank\ v < n \\ 0 \text{ if } otherwise \end{cases}$$

Meaning that 1 marks the valve to be open and 0 closed, respectively.

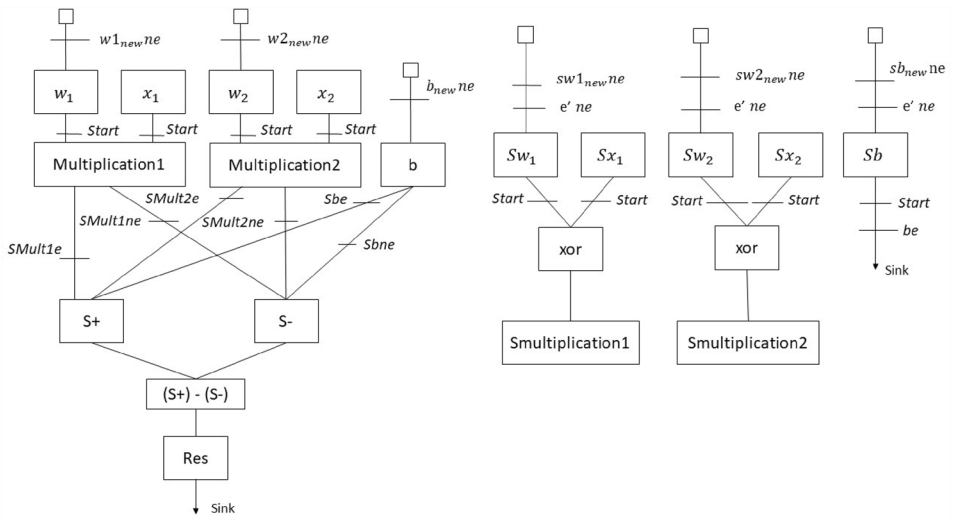


Fig. 4 Implementation 1: feed forward phase

$P = \{(reservoir, b, \{bnewne\}), (x1, S+, \{Smult1e, v\}),$
$(x1, S-, \{Smult1ne, v\}),$
$(x2, S+, \{Smult2e, v\}), (x2, S-, \{Smult2ne, v\}),$
$(b, S+, \{Sbe, v\}),$
$(b, S-, \{Sbne, v\}), (S+, [(S+) - (S-)], \{\}),$
$(S-, [(S+) - (S-)], \{\}),$
$([(S+) - (S-)], Res, \{\}), (Res, sink, \{\}),$
$(reservoir, Sw1, \{sw1newne, e'ne\}),$
$(reservoir, Sw2, \{sw2newne, e'ne\}),$
$(reservoir, Sb, \{sbnewne, e'ne\}),$
$(Sw1, xor, \{start\}), (Sx1, xor, \{start\}),$
$(Sw2, xor, \{start\}), (Sx2, xor, \{start\}),$
$(Sb, sink, \{start, be\}), (xor, Smultiplication1, \{\}),$
$(xor, Smultiplication2, \{\})\}$
$v_0 = \{(x1;0), (x2;0), (b;0),$
$(S+;0), (S-;0), (Res;0), (Sw1;0),$
$(Sx1;0), (Sw2;0), (Sx2;0), (Sb;0),$
$(Smultiplication1;0), (Smultiplication2;0),$
$(xor;0), ([(S+) - (S-)];0)\}$
$s_0 = \{(reservoir, b, \{bnewne\}, 0.1),$
$(x1, S+, \{Smult1e, v\}, 0.1),$
$(x1, S-, \{Smult1ne, v\}, 0.1),$
$(x2, S+, \{Smult2e, v\}, 0.1), (x2, S-, \{Smult2ne, v\}, 0.1),$
$(b, S+, \{Sbe, v\}, 0.1),$
$(b, S-, \{Sbne, v\}, 0.1), (S+, [(S+) - (S-)], \{\}, 0.1),$
$(S-, [(S+) - (S-)], \{\}, 0.1),$
$([(S+) - (S-)], Res, \{\}, 0.1), (Res, sink, \{\}, 0.1),$
$(reservoir, Sw1, \{sw1newne, e'ne\}, 0.1),$
$(reservoir, Sw2, \{sw2newne, e'ne\}, 0.1),$
$(reservoir, Sb, \{sbnewne, e'ne\}, 0.1),$
$(Sw1, xor, \{start\}, 0.1), (Sx1, xor, \{start\}, 0.1),$
$(Sw2, xor, \{start\}, 0.1), (Sx2, xor, \{start\}, 0.1),$
$(Sb, sink, \{start, be\}, 0.1), (xor, Smultiplication1, \{\}, 0.1),$
$(xor, Smultiplication2, \{\}, 0.1)\}$
$\Delta t = 1 second$

In the previous hypothesis, the connecting pipes between the tanks had the same diameter and did not affect the amount of water flowing through the network. To allow the passage of a "weighted" amount of water, a timer is used to regulate the opening of valves present in each connecting pipe between the nodes. Assuming that each second, an amount of water equal to the diameter of the pipe passes through each pipe, the water flow between the tanks can be controlled. The idea is to simplify the network by reducing the number of tanks and eliminating the multiplications involving the inputs and weights. (see Fig. 6)

To update the weights, we proceed as for the previous solution. In this case, however, the timer cannot be utilized to calculate the new weight, because we need the weight value in a tank in order to add it to the updated value.

To create a specific timer, the schema of Ring Oscillator presented in [10] is modified, by adding a valve between tank T1 and T2 (we have changed the name of the tanks to avoid confusion with the tanks representing the weights). The condition applied to the valve is $v \neq n$, which means that the valve remains open until the value n is reached in tank v (we also changed the name of tank y in v, to avoid confusion with tank y in the update weight phase). Additionally, in tank v, the tube and valve that allow water drainage are removed. This way, in each iteration, it is ensured that tank v will receive a quantity of water equal to 1, ideally assuming each iteration is 1 s (see Fig. 7).

4 Multilayer perceptron

In multilayer Perceptron, nodes have the same internal structure as for perceptron unit, but it is necessary to create new structure/schema for activation functions and backward propagation, and to modify the schema for updating weights.

Concerning activation functions, we have considered and implemented ReLu, Tanh, and Sigmoid.

4.1 ReLu

ReLu activation function is the simplest, for positive value the result is the input, for negative value the result is zero. The implementation consists of a tank, and a control valve so if the input is negative this valve is open and the volume of water in the tank is zero. (For keeping the amount of water small, it is possible to set a max limit, to do that we use a pipe set to a height value, if volume of water is over the value the water goes into the sink).

4.1.1 Tanh

Tanh activation function is more complex. We consider volume of water between 0 and 2. If the level of water is over 2, then we consider the corresponding value as 2. We separate the input volume of water into 3 tanks: in the first and second tank, the maximum value of water is 0.5, while in the third tank, it is 1. To separate the input water, we consider a water cascade like the one presented in [10], Fig. 1, and we operate on the volume of water of each tank in a different way. For the first tank, we take all the volume of water, (for example if it is 0.4 the result of tanh is 0.4). For second and third tank, we set an outflow pipe with a different diameter

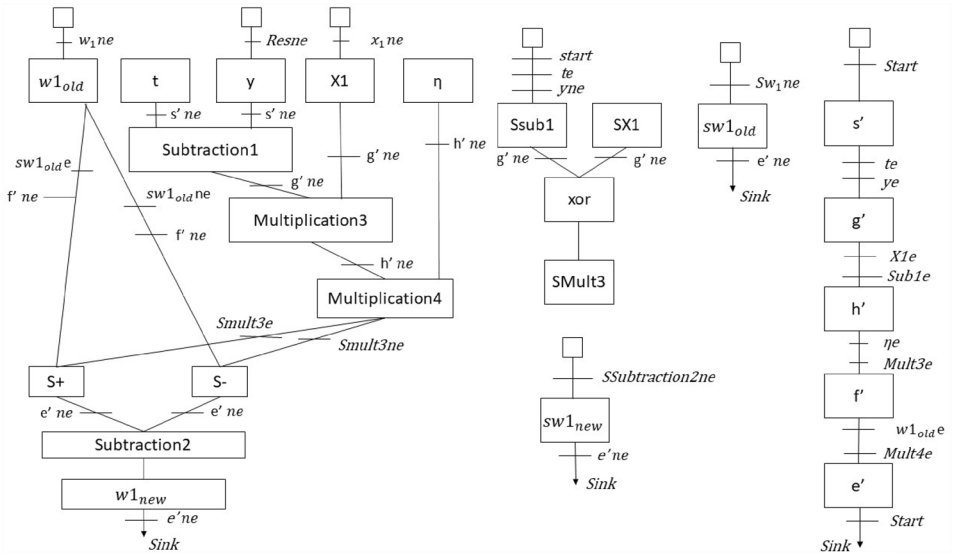


Fig. 5 Weight update phase

than 0.1 (default value), so for every second we have set a different outflow than 0.1, equal to the diameter of the pipe. This volume of water is drained into another tank that has two outflow pipes: one leads the water into the sink and the other into result tank, and these pipes have a different diameter than 0.1.

4.1.2 Sigmoid

For sigmoid function, we use the formula $(tanh(x/2) + 1)/2$.

We simply operate/compute the product (x*0.5) on the input of tanh, add 1 and perform another product on the result.

The activation functions just described can be compared with respect to three difference aspects:

- Accuracy with respect to the mathematical function they implement
- The time needed to provide a result
- Ease of implementation, i.e. the number of tanks and pipes required to implement them

Regarding ease of implementation, it appears that ReLU is the simplest, requiring only one tank and a valve. To implement the tanh, we need a total of 6 tanks and no valves, as the water cascade is exploited. While, for the sigmoid, we need to calculate the tanh value and, as before, we need 6 tanks initially, then 10 tanks for each multiplication, resulting in a total of 26 tanks plus 4 valves, 2 for each multiplication. The hyperbolic tangent (tanh) and sigmoid are therefore more complex to implement compared to ReLU. This complexity affects, of course, not only the number of tanks and valves but also, as a consequence, the speed at which results are obtained. In general, the more complex a system is, the more time it takes to complete operations and achieve a result. Nonetheless, by exploiting parallelism of this model, this can be partially avoided.

In fact, te computation time can be calculated on the basis of the number of levels of the various tanks. Tanks at level i communicate with tanks at level $i + 1$, in parallel. For example, considering Fig. 6, tanks (x1, x2, Sx1, Sx2, and reservoir) are at level zero, while tanks ([(S+)-(S-)], xor) are at level two.

The total time required to move from one level to the level below can be calculated by considering the maximum time required to let the flow of water from tanks at level i to tanks at level $i + 1$. This can be calculated by considering the maximum among all tanks, that is the maximum among the values

$$\frac{\text{volume of water in tank(i)}}{\text{diameter of pipe that exits from tank(i)}}$$

Regarding accuracy, ReLu stands out as the most accurate activation function, providing precise results without any significant approximation errors. Tanh, as defined, has a slight approximation error, while sigmoid suffers from significant errors due to the multiplication by the value 0.5. In conclusion, it becomes evident that ReLU outperforms both tanh and sigmoid functions in these three aspects.

4.2 Forward propagation

Forward propagation phase is the same as for perceptron forward propagation, but in a feedforward neural network structure, a unit will receive information of several units

belonging to the previous layer. We suppose to add all this information in one input tank for each node.

4.3 Backward propagation

In order to implement this phase, we first need to get the error value of the net. We use the formula

$$\delta = (t - y) * f'(y)$$

where $f'(y)$ is the first derivative, y is the output of the net, t is the label value. The idea for backward propagation is to rotate the net by 180 degrees, so that the output tank become the input of the net. In this tank, we put the error value, and then we propagate that value in the net like we did in forward propagation. In each node we get a delta value (that is, the error for each node); the delta values are used for updating the value of weights.

4.4 Update Weights

In this phase, we use the formula

$$w_{jk}^{new} = w_{jk}^{old} + n * \delta_k * y_j$$

where w_{jk} is the weight between the node k and node j, n is the learning rate, δ_k is the error calculated in the current node, and y_j is the output of upper layer node. The resulting schema is almost the same as perceptron updating weights schema.

5 Discussion and conclusions

In Implementation 1, the size of the network will be larger compared to that of Implementation 2, mainly due to the higher number of multiplications and tangent operations required.

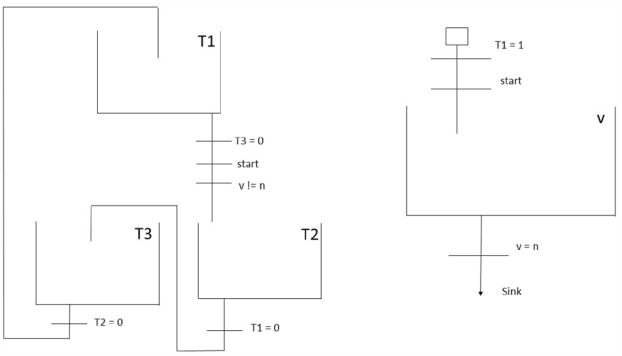


Fig. 7 Timer

To implement the timer that regulates the valves in Implementation 2, a pump will be needed to allow water to cycle through the three tanks. In a future implementation, the effects of the pump on the network will also need to be taken into account.

Regarding training, the network in Implementation 2 requires that the connecting tubes between the various nodes be changed with every weight update. This will surely make the training process of the network much harder with respect to Implementation 1.

Summarizing, the network in Implementation 2 requires a smaller number of tanks, tubes, and multiplication structures: as a consequence, it is the simplest to implement in terms of size. On the contrary, the network in Implementation 1 is the simplest to train, as it does not require modifying the dimension of the tubes every time the weight values change.

Future investigations concern the tests that could be conducted: for example, using more precise learning rates (e.g. vary in hundredths rather than just decimal places), to check whether or not better results can be obtained.

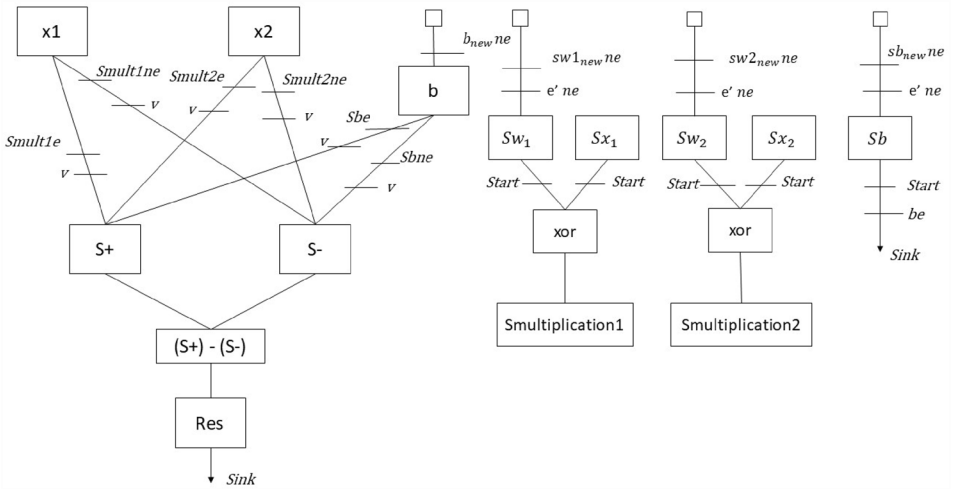


Fig. 6 Implementation 2: Feed Forward phase

Studies on more complex networks such as RNN (Recurrent Neural Network) or LSTM (Long Short-Term Memory) are another important research direction.

Acknowledgements The work of Claudio Zandron was partially supported by Università degli Studi di Milano-Bicocca, Fondo Ateneo per la Ricerca 2023, project 2023-ATE-0333.

Author Contributions N.C. proposed main idea behind this work. N.C. prepared all figures. All authors contributed to the research, and to write and review the manuscript.

Funding Open access funding provided by Università degli Studi di Milano - Bicocca within the CRUI-CARE Agreement.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Alhazov, A., Leporati, A., Manzoni, L., Mauri, G., & Zandron, C. (2021). Alternative space definitions for P systems with active membranes. *Journal of Membrane Computing, 3*, 87–96. https://doi.org/10.1007/s41965-021-00074-2
2. Ballesteros, K. J., Cailipan, D. P. P., de la Cruz, R. T. A., Cabarle, F. G. C., & Adorna, H. N. (2022). Matrix representation and simulation algorithm of numerical spiking neural p systems. *Journal of Membrane Computing, 4*(1), 41–55.
3. Baquero, F., Campos, M., Llorens, C., & Sempere, J. (2021). P systems in the time of COVID-19. *Journal of Membrane Computing, 3*, 246–257. https://doi.org/10.1007/s41965-021-00083-1
4. Battyányi, P., & Vaszil, G. (2020). Description of membrane systems with time Petri nets: Promoters/inhibitors, membrane dissolution, and priorities. *Journal of Membrane Computing, 2*, 341–354. https://doi.org/10.1007/s41965-020-00062-y
5. Ciencialová, L., Csuhaj-Varjú, E., Cienciala, L., et al. (2019). P colonies. *Journal of Membrane Computing, 1*, 178–197. https://doi.org/10.1007/s41965-019-00019-w
6. de la Cruz, R. T. A., Cabarle, F. G. C., Macababayao, I. C. H., et al. (2021). Homogeneous spiking neural P systems with structural plasticity. *Journal of Membrane Computing, 3*, 10–21. https://doi.org/10.1007/s41965-020-00067-7
7. Díaz-Pernil, D., Gutierrez-Naranjo, M. A., & Peng, H. (2019). Membrane computing and image processing: A short survey. *Journal of Membrane Computing, 1*, 58–73. https://doi.org/10.1007/s41965-018-00002-x
8. Dupaya, A. G. S., Galano, A. C. A. P., Cabarle, F. G. C., De La Cruz, R. T., Ballesteros, K. J., & Lazo, P. P. L. (2022). A web-based visual simulator for spiking neural p systems. *Journal of Membrane Computing, 4*(1), 21–40.
9. Gheorghe, M., Lefticaru, R., Konur, S., Nicolescu, I., & Adorna, H. N. (2021). Spiking neural P systems: Matrix representation and formal verification. *Journal of Membrane Computing, 3*, 133–148. https://doi.org/10.1007/s41965-021-00075-1
10. Hinze, T., Happe, H., Henderson, A., & Nicolescu, R. (2020). Membrane Computing with Water. *Journal of Membrane Computing, Springer, 2*, 121–136.
11. Henderson, A., Nicolescu, R., Dinneen, M. J., Chan, T., Happe, H., & Hinze, T. (2021). Turing Completeness of Water Computing. *Journal of Membrane Computing, Springer, 3*(3), 182–193.
12. Henderson, A., Nicolescu, R., Dinneen, M. J., Chan, T., Happe, H., & Hinze, T. (2023). Programmable and Parallel Water Computing. *Journal of Membrane Computing, Springer, 5*(1), 25–54.
13. Ionescu, M., Păun, Gh., & Yokomori, T. (2006). Spiking neural P systems. *Fundamenta Informaticae, 71*(2,3), 279–308.
14. Langer, M., & Valenta, D. (2023). On evolving environment of 2D P colonies: Ant colony simulation. *Journal of Membrane Computing, 5*, 117–128. https://doi.org/10.1007/s41965-023-00123-y
15. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2017). Characterising the complexity of tissue P systems with fission rules. *Journal of Computer and System Sciences, 90*, 115–128.
16. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2019). Characterizing PSPACE with shallow non-confluent P systems. *Journal of Membrane Computing, 1*, 75–84. https://doi.org/10.1007/s41965-019-00011-4
17. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2020). Shallow laconic P systems can count. *Journal of Membrane Computing, 2*, 49–58. https://doi.org/10.1007/s41965-020-00032-4
18. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2020). A Turing machine simulation by P systems without charges. *Journal of Membrane Computing, 2*, 71–79. https://doi.org/10.1007/s41965-020-00031-5
19. Leporati, A., Mauri, G., & Zandron, C. (2022). Spiking neural P systems: Main ideas and results. *Natural Computing, 21*(4), 629–649.
20. Liu, Y., Nicolescu, R., & Sun, J. (2020). Formal verification of cP systems using PAT3 and ProB. *Journal of Membrane Computing, 2*(2), 80–94.
21. Liu, Y., Nicolescu, R., & Sun, J. (2021). Formal verification of cP systems using Coq. *Journal of Membrane Computing, 3*(3), 205–220.
22. Lv, Z., Yang, Q., Peng, H., et al. (2021). Computational power of sequential spiking neural P systems with multiple channels. *Journal of Membrane Computing, 3*, 270–283. https://doi.org/10.1007/s41965-021-00089-9
23. Martín-Vide, C., Păun, Gh., Pazos, J., & Rodriguez-Paton, A. (2003). Tissue P systems. *Theoretical Computer Science, 296*(2), 295–326.
24. McCulloch, W., & Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*(4), 115–133. https://doi.org/10.1007/BF02478259
25. Nadizar, G., & Pietropolli, G. (2023). A grammatical evolution approach to the automatic inference of P systems. *Journal of Membrane Computing, 5*, 129–143.
26. Orellana-Martín, D., Valencia-Cabrera, L., Riscos-Núñez, A., & Perez-Jimenez, M. J. (2019). P systems with proteins: A new frontier when membrane division disappears. *Journal*

of Membrane Computing, 1, 29–39. https://doi.org/10.1007/s41965-018-00003-w
27. Păun, Gh. (2000). Computing with membranes. *Journal of Computer and System Sciences, 61*(1), 108–143.
28. Păun, Gh. (2001). P systems with active membranes: Attacking NP-Complete problems. *Journal of Automata, Languages and Combinatorics, 6*(1), 75–90.
29. Păun, Gh., Rozenberg, G., & Salomaa, A. (Eds.). (2010). *The Oxford Handbook of Membrane Computing*. New York: Oxford University Press.
30. Pérez-Hurtado, I., Orellana-Martín, D., Zhang, G., et al. (2019). P-Lingua in two steps: Flexibility and efficiency. *Journal of Membrane Computing, 1*, 93–102. https://doi.org/10.1007/s41965-019-00014-1
31. Qiu, C., Xue, J., Liu, X., et al. (2022). Deep dynamic spiking neural P systems with applications in organ segmentation. *Journal of Membrane Computing, 4*, 329–340.
32. Sempere, J. M. (2023). Modeling Markov sources and hidden Markov models by P systems. *Journal of Membrane Computing, 5*, 161–169. https://doi.org/10.1007/s41965-023-00129-6
33. Sosik, P. (2019). P systems attacking hard problems beyond NP: A survey. *Journal of Membrane Computing, 1*, 198–208.
34. Sosík, P., Drastík, J., Smolka, V., et al. (2020). From P systems to morphogenetic systems: An overview and open problems. *Journal of Membrane Computing, 2*, 380–391. https://doi.org/10.1007/s41965-020-00057-9
35. Turlea, A., Gheorghe, M., Ipate, F., & Konur, S. (2019). Search-based testing in membrane computing. *Journal of Membrane Computing, 1*, 241–250. https://doi.org/10.1007/s41965-019-00027-w
36. Valencia-Cabrera, L., Perez-Hurtado, I., & Martinez-del Amor, M. A. (2020). Simulation challenges in membrane computing. *Journal of Membrane Computing, 2*, 1–11.
37. Verlan, S., Freund, R., Alhazov, A., et al. (2020). A formal framework for spiking neural P systems. *Journal of Membrane Computing, 2*, 355–368. https://doi.org/10.1007/s41965-020-00050-2
38. Yu, W., Wu, J., Chen, Y., et al. (2023). Fuzzy tissue-like P systems with promoters and their application in power coordinated control of microgrid. *Journal of Membrane Computing, 5*, 1–11. https://doi.org/10.1007/s41965-022-00109-2
39. Yu, W., Xiao, X., Wu, J., et al. (2023). Application of fuzzy spiking neural dP systems in energy coordinated control of multi-microgrid. *Journal of Membrane Computing, 5*, 69–80. https://doi.org/10.1007/s41965-023-00118-9
40. Zhao, S., Zhang, L., Liu, Z., et al. (2022). ConvSNP: a deep learning model embedded with SNP-like neurons. *Journal of Membrane Computing, 4*, 87–95.