Contemporary Mathematics
http://ojs.wiserpub.com/index.php/CM/

Research Article

Partial Structural Equivalence Between Integral Neurons and a Subclass of RHONN Neurons

Kostadin Yotov, Emil Hadzhikolev, Stanka Hadzhikoleva*

Faculty of Mathematics and Informatics, University of Plovdiv Paisii Hilendarski, Plovdiv, Bulgaria
E-mail: stankah@uni-plovdiv.bg

Received: 29 December 2025; **Revised:** 21 April 2026; **Accepted:** 30 April 2026

Abstract: This paper presents a comparative theoretical and experimental study of two contemporary neuron models: the integral neuron and the Recurrent High-Order Neural Networks (RHONN) neuron. The integral neuron forms its internal signal through an integral aggregation of the input stimuli modulated by a continuous weight function, whose form can be adapted to the requirements of a given problem. In contrast, the RHONN neuron employs high-order products of the input variables, enabling the modeling of complex and nonlinear interactions among inputs. The analysis examines structural properties, computational complexity, robustness to noise, and behavior under varying data volumes and input dimensionality. A partial structural equivalence is established between an integral neuron with a polynomial weight function and a specific subclass of RHONN neurons with symmetric interactions of the corresponding order. Furthermore, the integral neuron is shown to exhibit significantly lower computational complexity during training, as well as increased robustness in scenarios involving limited data and noisy inputs. The results indicate that, while the two neuron models possess distinct expressive capabilities, the integral neuron provides a compact and robust alternative to RHONN neurons for a broad class of approximation and learning tasks, particularly in high-dimensional or noise-contaminated settings.

Keywords: integral neuron, Recurrent High-Order Neural Networks (RHONN), High-Order Neural Networks (HONNs), structural equivalence, computational complexity, robustness to noise, curse of dimensionality, function approximation

MSC: 68T15, 68T05

1. Introduction

Recurrent High-Order Neural Networks (RHONNs) are a class of Artificial Neural Networks (ANNs) that incorporate high-order interactions among input data and internal states. Giles and Maxwell [1] developed the theory of High-Order Neural Networks (HONNs), to which recurrent connections were later added. One of the earliest academic publications in which RHONNs were formalized and applied to control tasks is [2].

RHONNs extend traditional Recurrent Neural Networks (RNNs) by incorporating nonlinear interactions that can capture more complex dependencies in sequences [3–6]. While standard RNN models use linear combinations of the previous state and the input, RHONNs add terms that are products of multiple inputs or internal states. This increases the model's capacity to capture complex relationships, as the high-order terms enable the network to handle nonlinear dependencies that are difficult to model with traditional RNNs. The output of the j-th neuron in a RHONN is given by:

Copyright ©2026 Stanka Hadzhikoleva, et al.
DOI: https://doi.org/10.37256/cm.7420269503
This is an open-access article distributed under a CC BY license
(Creative Commons Attribution 4.0 International License)
https://creativecommons.org/licenses/by/4.0/

$$y_j(t) = g\left(\sum_{i=1}^{n}\sum_{p=1}^{D} w_{ij}^{(h)} \prod_{q=1}^{p} x_{iq}(t) + \sum_{k=1}^{m} \alpha_{jk} y_k(t-1) + b_j\right). \tag{1}$$

where:

- $x_{iq}(t)$ is the q-th input signal at time t;
- $w_{ij}^{(p)}$ is the weight associated with a p-th order combination of inputs;
- D is the maximum order of interaction among the inputs;
- $\prod_{q=1}^{p} x_{iq}(t)$ denotes the product of p input signals corresponding to a p-th order term;
- $y_k(t-1)$ is the output of the k-th neuron at the previous time instant;
- α_{jk} is the feedback weight from the k-th neuron;
- b_j is the bias term of the j-th neuron;
- g is the transfer (activation) function.

In addition to the fact that they capture nonlinear dependencies through high-order interactions that standard RNNs cannot process, RHONNs are suitable for a wide range of tasks, including time-series analysis [7–10], text processing [11–14], biological signal analysis [15–18], and dynamical systems modeling [19–22]. Moreover, they exhibit high expressiveness: it is precisely the high-order terms that extend the network's ability to model complex dependencies.

Despite these advantages, their limitations must also be noted. The inclusion of high-order interactions increases the number of parameters and, consequently, the computational and memory requirements. The large parameter space makes RHONNs prone to overfitting, especially when trained on small datasets, and the associated optimization algorithms become more complex and require careful tuning, particularly for high interaction orders.

In contrast to standard RNNs, which operate on linear combinations of inputs and internal states, RHONNs extend the model by incorporating high-order interactions. This makes them more expressive, but also significantly more resource-intensive.

On the other hand, in [23] the notions of a weight function and an integral neuron are defined as follows.

Let C^k denote the class of k -times differentiable functions, and let $\mathbb{N}_0$ be the set of natural numbers augmented by zero, i.e., $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Definition 1 (Neuron weight function) A continuously differentiable real-valued function $w(t) \in C^k$, $k \in \mathbb{N}_0$, which defines the aggregated signal in the body of an artificial neuron through the relation

$$S = \int_0^{\Sigma_{i=1}^{n} x_i} w(t)dt, \tag{2}$$

where $x_1, x_2, \ldots, x_n$ are the input stimuli, is called a weight function of the corresponding neuron.

Definition 2 (Integral neuron) An artificial neuron that uses a weight function $w(t) \in C^k$, $k \in \mathbb{N}_0$, to form the signal in its body is called an integral neuron of class $(k+1)$.

According to the manner in which the new structure is defined, it is characterized by a nonlinear integral form of the signal in its

$$S = \int_0^{\Sigma_{i=1}^{n} x_i} w(t)dt \tag{3}$$

and the output signal is determined by

$$Out = g\left(\int_0^{\Sigma_{i=1}^n x_i} w(t)dt\right), \tag{4}$$

where $w(t)$ and g are the weight function and the transfer (activation) function of the neuron, respectively (Figure 1).

Integral neurons are distinguished by the fact that, instead of traditional fixed weights, they employ a weight function $w(t)$, which not only replaces the weights but also imparts a nonlinear character to the signal in the neuron body. Owing to this function, the optimization algorithms are no longer aimed at finding static weights, but rather at determining the optimal shape of the function $w(t)$ by optimizing its parameters so as to achieve minimal error. Empirical and theoretical studies show that algorithms such as gradient descent, Gauss-Newton methods, and the generalized Levenberg-Marquardt algorithm exhibit good convergence properties when training integral neurons. It has been successfully established that the following theorems hold.

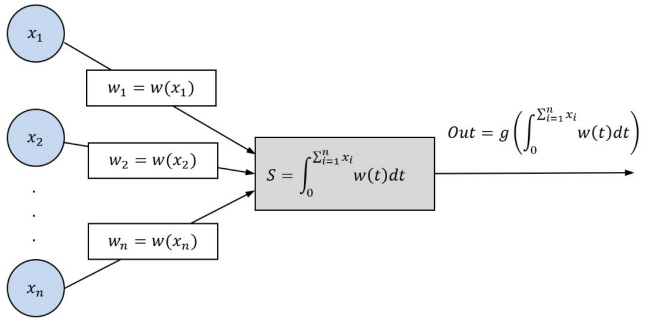


Figure 1. Integral neuron equipped with smooth weight and transfer functions, $w(t)$ and g, respectively

Theorem 1 (Convergence of the gradient descent method in optimizing the weight function) Let an integral neuron of at least first class be given, equipped with a smooth transfer function g. Let a denote the vector containing the parameters of the weight function $w(t)$, and $\frac{\partial E}{\partial w(t)}$ be the gradient of the error function.

Then, if the update of the parameters of $w(t)$ is performed by an optimization algorithm based on gradient descent applied to the mean squared error function $E(w(t))$, with a step size $\eta > 0$,

$$a^{(k+1)}(t) = a^k(t) - \eta \frac{\partial E}{\partial w(t)}, \tag{5}$$

the algorithm is convergent and drives the parameters of the function $w(t)$ toward values that minimize $E(w(t))$.

Theorem 2 (Convergence of the Gauss-Newton method in optimizing the weight function) Let an integral neuron of at least second class be given, equipped with a smooth transfer function g. Let a denote the vector containing the parameters of the weight function, J the Jacobian matrix of the error function $E(w(t))$, and e the error vector defined by

$$e_j = h_j - g\left(\int_0^{\Sigma_{i=1}^n x_{ji}} w(t)dt\right), \; j = 1, 2, \ldots, m. \tag{6}$$

Then, if the Jacobian matrix J has full rank and the update of the parameters of $w(t)$ is performed using the Gauss–Newton optimization method,

$$a^{(k+1)} = a^{(k)} - \left(J^T J\right)^{-1} J^T e, \tag{7}$$

the algorithm is convergent and drives the parameters of the function $w(t)$ toward values that minimize $E\left(w(t)\right)$.

Theorem 3 (Convergence of the Levenberg-Marquardt method in optimizing the weight function) Let an integral neuron of at least second class be given, equipped with a smooth transfer function g. Let a denote the vector containing the parameters of the weight function, J the Jacobian matrix of the error function $E\left(w(t)\right)$, and e the error vector defined by

$$e_j = h_j - g\left(\int_0^{\Sigma_{i=1}^n x_{ji}} w(t)dt\right),\ j = 1,\ 2,\ \ldots,\ m. \tag{8}$$

Let $\lambda > 0$ be the damping parameter that controls the transition between the gradient descent method ($\lambda \to \infty$) and the Gauss-Newton method ($\lambda \to 0$), and let I denote the identity matrix of the same order as J.

Then, if the Jacobian matrix J has full rank and the update of the parameters of $w(t)$ is performed using the Levenberg-Marquardt optimization method,

$$a^{(k+1)} = a^{(k)} - \left(J^T J + \lambda I\right)^{-1} J^T e, \tag{9}$$

the algorithm is convergent and drives the parameters of the function $w(t)$ toward values that minimize $E\left(w(t)\right)$.

In all three theorems,

$$a = \left[a_0,\ a_1,\ a_2,\ \ldots, a_q\right]^T, \tag{10}$$

and

$$\frac{\partial E}{\partial w(t)} = \left[\frac{\partial E}{\partial a_0},\ \frac{\partial E}{\partial a_1},\ \frac{\partial E}{\partial a_2}, \ldots,\ \frac{\partial E}{\partial a_q}\right]^T. \tag{11}$$

are, respectively, the parameter vector of the weight function and the gradient of the error function.

J is the Jacobian matrix containing the partial derivatives of the error $E\left(w(t)\right)$ with respect to each parameter of $w(t)$:

$$J = \begin{bmatrix} \frac{\partial e_1}{\partial a_0} & \frac{\partial e_1}{\partial a_1} & \cdots & \frac{\partial e_1}{\partial a_q} \\ \frac{\partial e_2}{\partial a_0} & \frac{\partial e_2}{\partial a_1} & \cdots & \frac{\partial e_2}{\partial a_q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_m}{\partial a_0} & \frac{\partial e_m}{\partial a_1} & \cdots & \frac{\partial e_m}{\partial a_q} \end{bmatrix}_{m\times(q+1)}. \tag{12}$$

Moreover,

$$e = [e_1,\ e_2,\ e_3,\ \ldots, e_m]^T \tag{13}$$

is the vector that contains all residual errors obtained when all training samples are propagated through the neuron at each iteration, where m is the number of training samples and $q+1$ is the number of parameters of the weight function.

$H \approx J^T J$ is the Hessian matrix and can be written as

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial a_0^2} & \frac{\partial^2 E}{\partial a_0 \partial a_1} & \cdots & \frac{\partial^2 E}{\partial a_0 \partial a_q} \\ \frac{\partial^2 E}{\partial a_1 \partial a_0} & \frac{\partial^2 E}{\partial a_1^2} & \cdots & \frac{\partial^2 E}{\partial a_1 \partial a_q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial a_q \partial a_0} & \frac{\partial^2 E}{\partial a_q \partial a_1} & \cdots & \frac{\partial^2 E}{\partial a_q^2} \end{bmatrix}_{(q+1)\times(q+1)} . \tag{14}$$

Considering the characteristics of the two neurons discussed above, the hypothesis of a possible functional similarity between them naturally arises.

A RHONN is used to process complex input dependencies through polynomial interactions among the inputs and through feedback connections. Its output, or an internal layer, can be interpreted as a function that transforms the input vector into a scalar or into a lower-dimensional representation before it is used, for example, for prediction. In other words, a RHONN can be viewed as a function of the input that summarizes the salient characteristics of the signal.

In turn, the integral neuron can be viewed as a block that transforms the input vector into a scalar quantity via an integral: $S = \int_0^{f(x)} w(t)dt$, where $f(x) = \sum_{i=1}^{n} x_i$. In this way, the entire multidimensional input is aggregated and projected onto a single scalar value, which is then passed to the transfer function.

This observed similarity between the two neurons motivates a systematic comparison between them. To what extent do the two neuron models resemble each other, and in what respects do they differ? Are there domains in which they overlap and can be considered interchangeable? These questions provide the basis for formulating the following research question:

Do there exist specific conditions under which the integral neuron and the RHONN can be functionally equivalent, such that the integral model preserves key characteristics of the RHONN—such as high expressiveness and adaptability—while achieving a more compact architecture and lower computational cost?

Based on the mathematical models of the two neuron types, several differences can be identified at an early stage, as summarized in Table 1.

From this comparison, it follows that while RHONNs emphasize high-order interactions and recurrence, integral neurons introduce a different paradigm based on continuous weight functions and a more compact architectural representation.

Preliminary analysis suggests that the integral neuron may offer advantages in terms of computational efficiency, robustness to noise, and stability under limited data, while RHONNs retain strengths in modeling localized and non-symmetric interactions.

In contrast to existing studies on RHONNs, which primarily focus on high-order interactions and recurrent dynamics, and in contrast to our previous work on integral neurons, which introduced the use of continuous weight functions for signal aggregation, the present study establishes an explicit theoretical and experimental connection between these two neuron models.

Table 1. Comparison of the main characteristics of RHONN and the integral neuron

Characteristic	RHONN	Integral neuron
Weight parameterization	They employ weights $w_{ij}^{(p)}$, which are fixed for a given combination of inputs and interaction orders. These weights are associated with combinations of input signals and their higher-order terms (products, polynomials). They are scalar values optimized during training.	The weights are expressed as functions $w(t)$ whose shape is determined by a set of parameters. This weight function defines the nonlinear processing of the inputs through integration, allowing the modeling of more complex dependencies.
Feedback	They include explicit feedback through the term $\sum_{k=1}^{m} \alpha_{jk} y_k(t-1)$, which makes them suitable for processing temporal sequences.	There is no explicit feedback mechanism. The processing is feedforward, and the signal in the neuron body depends solely on the integral of the inputs.
Order of interaction	They support higher-order interactions through products of the form $\prod_{q=1}^{h} x_{iq}(t)$.	The order of interaction depends on the structure of the weight function $w(t)$. For example, if $w(t)$ is a polynomial, the interaction order can be controlled by the degree of the polynomial.
Architectural complexity	The complexity of a RHONN grows exponentially with the interaction order D, since the number of weights$w_{ij}^{(p)}$ increases significantly.	The complexity is governed by the number of parameters of $w(t)$, which enables a more compact representation.

To the best of our knowledge, the relationship between integral neurons and RHONN architectures has not been explicitly established or formally analyzed in the existing literature. In particular, we demonstrate that an integral neuron with a polynomial weight function admits a partial structural equivalence with a specific subclass of RHONN neurons under clearly formulated conditions.

Our analysis provides a systematic evaluation of the two models with respect to computational complexity, robustness to noise, behavior under limited training data, and sensitivity to increasing input dimensionality. These results clarify both the shared structural foundation and the essential differences between the two architectures, positioning the integral neuron not merely as an alternative construction, but as a compact and stable modeling framework with partially overlapping expressive capabilities.

The main contributions of this work can be summarized as follows:

• We establish a partial structural equivalence between an integral neuron with a polynomial weight function and a specific subclass of RHONN neurons characterized by symmetric aggregated interactions of the corresponding order.

• We prove that the sets of signals generated by integral neurons and RHONN neurons have a non-empty intersection, while neither class is fully contained within the other.

• We provide a systematic comparative analysis of the two models with respect to representational structure, computational complexity, robustness to noise, behavior under limited training data, and sensitivity to input dimensionality.

• We demonstrate experimentally that the integral neuron offers a more compact and computationally efficient alternative in several practically relevant regimes, while RHONN models retain advantages in capturing localized and non-symmetric interactions among inputs.

The motivation for this study arises from the increasing computational and statistical limitations of RHONN-based models. Although highly expressive, RHONNs become progressively more expensive and difficult to train as the interaction order and input dimensionality increase. Their large parameter space also makes them prone to overfitting, particularly in the presence of limited or noisy training data.

In contrast, the integral neuron provides a compact and continuous modeling framework, in which nonlinear processing is governed by a weight function. This suggests the possibility of achieving comparable representational power at a lower computational cost and with improved robustness.

The remainder of the paper is organized as follows. Section 2 presents a comparative analysis of the computational complexity of RHONN and integral neurons. Section 3 examines the influence of input dimensionality and training data volume. Section 4 investigates the robustness of the two models to noise in different settings. Section 5 summarizes the comparative findings, while Section 6 establishes the theoretical results concerning the partial structural equivalence and the intersection of the signal spaces generated by the two neuron models.

2. Computational complexity of RHONN neurons and integral neurons

The computational complexity of training algorithms affects the training and prediction time, as well as memory usage and processor workload. These characteristics are of critical importance under limited computational resources and when faster training and inference are required.

In the context of growing concerns about sustainability and energy efficiency, the choice of more compact and lightweight models has a direct impact on the energy consumption of computing devices. Large neural networks—especially recurrent and high-order ones such as RHONNs—require substantial computational resources, which leads to higher power consumption, particularly during long training processes or when deployed on battery-constrained devices. In this respect, integral neurons, with their smaller number of parameters and lower computational complexity, offer a more sustainable alternative, especially in systems where energy efficiency is critical. This aligns with the increasing demand for "green" technologies in artificial intelligence, where an optimal balance between computational power and environmental impact is sought.

In this section, an analysis of the computational complexity of RHONN neurons and integral neurons is presented for training using the Levenberg-Marquardt algorithm.

To ensure a fair comparison, we analyze the computational complexity of an RHONN layer consisting of r output neurons and compare it with the training complexity of a single integral neuron. This choice is motivated by the fact that an integral neuron aggregates all input dimensions into a scalar internal signal and thus plays a role analogous to that of an entire RHONN layer with respect to input data aggregation.

2.1 *Complexity of the Levenberg-Marquardt algorithm for RHONN models*

A RHONN neuron involves the training of a large number of parameters, since it incorporates high-order interactions among the inputs and additionally employs feedback connections from previous time steps. The main parameters of the RHONN architecture are as follows:

- m: number of training samples (time points);
- n: number of inputs;
- D: maximum order of interaction among the inputs;
- r: number of output neurons in the layer (instead of a single one);
- k: number of iterations until convergence.

The number of weights $w_{ij}^{(q)}$, for each input i, interaction order q, and neuron j, is given by

$$r \sum_{q=1}^{D} \binom{n}{q} \approx rO(n^D). \tag{15}$$

The number of feedback weights (α_{jk}) is $r \cdot r = r^2$, and the number of bias terms is r. Thus, the approximate total number of parameters is

$$h = rO\left(n^D\right) + r^2 + r \approx rn^D \quad (dominant\ term). \tag{16}$$

Let us consider the complexity of the individual operations within a single iteration.

- Error computation: For each time instant t (respectively, each training sample), the output of each neuron is computed, which involves all products up to order D, i.e., $O(n^D)$ operations per neuron. Taking into account the number of training samples and the number of output neurons, this yields $O(m \cdot r \cdot n^D)$.
- Computation of the Jacobian J: The Jacobian has dimension $m \times h = m \times r \cdot n^D$, which implies a complexity of $O(m \cdot r \cdot n^D)$.

• Matrix multiplication $J^T J$: The resulting matrix has size $h \times h = (r \cdot n^D)^2$. Hence, the per-iteration complexity here is $O\left(m \cdot (r \cdot n^D)^2\right) = O(m \cdot r^2 \cdot n^{2D})$.

• Inversion of $J^T J + \lambda I$: $O\left((r \cdot n^D)^3\right) = O(r^3 \cdot n^{3D})$.

• Matrix–vector product $J^T e$: $O(m \cdot r \cdot n^D)$.

Therefore, the overall complexity per iteration, dominated by the most expensive operations for k iterations, is

$$O\left(k\left(m \cdot r^2 \cdot n^{2D} + r^3 \cdot n^{3D}\right)\right). \tag{17}$$

2.2 *Complexity of the Levenberg-Marquardt algorithm for an integral neuron*

The computational complexity of the Levenberg-Marquardt (LM) algorithm for optimizing the weight function $w(t)$ can be analyzed by examining the main operations performed in each iteration and their contribution to the overall computational load.

Main operations in the algorithm are:

• Parameter update: The transition

$$a^{(k+1)} = a^{(k)} - \left(J^T J + \lambda I\right)^{-1} J^T e \tag{18}$$

involves computing the Jacobian J, matrix-matrix multiplication, transposition, matrix inversion, and matrix-vector operations.

• Error computation: For each training sample, the mean squared error E is evaluated,

$$E\left(w(t)\right) = \sum_{j=1}^{m} \left(h_j - g\left(\int_0^{\Sigma_{i=1}^{n} x_{ji}} w(t)dt\right)\right)^2, \tag{19}$$

which requires processing all input data and evaluating the integral involving $w(t)$.

Let m denote the number of training samples. The residual error vector e in (18) is computed for each sample, so the number of computations depends linearly on m. Let p denote the number of parameters of the weight function $w(t)$ (e.g., polynomial coefficients). Then the Jacobian J is an $m \times p$, and $J^T J$ has dimension $p \times p$.

Thus, the computational complexity of the operations in (18) is as follows:

• Computation of J: $O(m \cdot p)$, since evaluating the derivatives scales with the number of parameters and the number of samples.

• Matrix multiplication $J^T J$: $O\left(m \cdot p^2\right)$.

• Inversion of $J^T J + \lambda I$: $O(p^3)$, assuming standard matrix inversion algorithms.

• Matrix-vector multiplication $J^T e$: $O(m \cdot p)$.

Here we present the asymptotic complexity by retaining only the dominant terms. Therefore, for k iterations required to reach convergence, the overall complexity is given by:

$$O\left(k\left(m \cdot p^2 + p^3\right)\right). \tag{20}$$

However, the following remark should also be made. At first glance, one might get the impression that the number of inputs does not contribute to the computational complexity. It is therefore important to note that, in the complexity

analysis of the Levenberg-Marquardt algorithm for the integral neuron, the number of inputs n is implicitly included in the computations but is treated as part of the parameter count p (e.g., the number of coefficients of the weight function $w(t)$ and, in practical implementations, also through the cost of evaluating the integral for each of the m training samples).

Let us examine the relationship between the number of inputs and the number of parameters p.

For an integral neuron, the internal signal S is computed as

$$S = \int_0^{\sum_{i=1}^n x_i} w(t)dt, \tag{21}$$

where n is the number of inputs.

In the complexity analysis, the computation of $\sum_{i=1}^n x_i$ is treated as part of the "preprocessing operations" and is assumed to contribute a negligible overhead relative to the dominant optimization computations. It requires one summation over n addends. The parameters of the weight function $w(t)$ are the main focus, since they are the quantities being optimized, whereas the number of inputs n does not directly affect the number of these parameters.

On the other hand, the integral in (21) is evaluated numerically for each training sample. In numerical integration, the complexity depends on the number of discretization steps rather than directly on the number of inputs n. Typically, this discretization cost is treated as constant with respect to n. In the Jacobian matrix J, the rows correspond to the m training samples and the columns correspond to the p parameters of the weight function. The number of inputs n does not directly influence the dimensionality of J, because the inputs are aggregated into the integral signal S.

Finally, one should account for the possibility of a local, indirect dependence. If the number of inputs n influences the chosen parameterization of the weight function $w(t)$ (e.g., a larger number of inputs motivates selecting a larger p), then n affects the complexity indirectly via an increase in the number of parameters p. However, this occurs a priori; once p is fixed, the subsequent computational complexity is governed by the resulting parameter count of the weight function.

2.3 *Example: comparing computational complexity*

In this section, we compare the computational complexity of the RHONN architecture (which can be reduced to the case of a single output for $r = 1$) with that of a single integral neuron, reflecting their respective roles as aggregation mechanisms.

Before continuing the comparison of the computational complexity of the two neuron models, let us consider the comparative Table 2.

Table 2. Comparative table of the computational complexity of the RHONN layer and the integral neuron

Model	Number of parameters p	Complexity per iteration	Total complexity for k iterations
RHONN layer	$h = r \cdot n^D$	$O\left(m \cdot r^2 \cdot n^{2D} + r^3 \cdot n^{3D}\right)$	$O\left(k(m \cdot r^2 \cdot n^{2D} + r^3 \cdot n^{3D})\right)$
Integral neuron	p – number of parameters of $w(t)$	$O\left(m \cdot p^2 + p^3\right)$	$O\left(k\left(m \cdot p^2 + p^3\right)\right)$

It is evident that RHONN architecture exhibit computational complexity that grows exponentially with respect to the number of inputs n and the interaction order D, whereas the integral neuron has complexity that grows polynomially with respect to the number of parameters of the weight function. This makes RHONNs more flexible for modeling temporal dependencies, but also substantially more resource-intensive for large n and D (Figure 2).

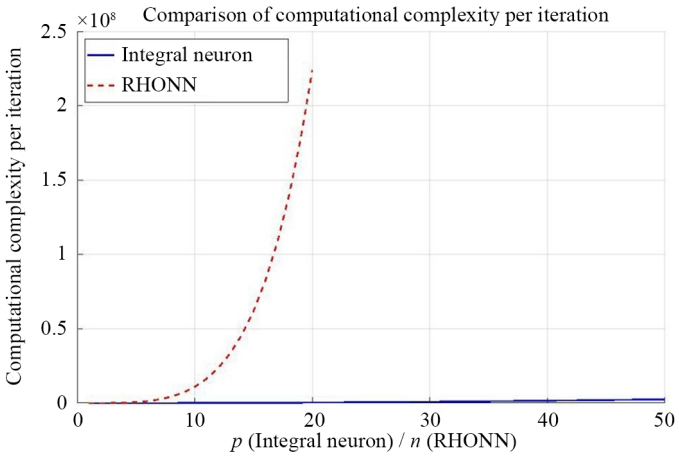


Figure 2. Comparison of the computational complexity of the two models for one iteration as the number of parameters increases. In the example, the interaction order is fixed at $D = 2$ (MATLAB 2018a)

It should nevertheless be noted that the per-iteration computational complexity for a RHONN layer changes not only with the number of inputs n, but also with the interaction order D:

$$C_{RHONN}(n,D) = O\left(m \cdot r^2 \cdot n^{2D} + r^3 \cdot n^{3D}\right). \tag{22}$$

In other words, the complexity is a two-dimensional function. This motivates considering a 3D surface (Figure 3), which illustrates how sharply the complexity increases when both parameters grow simultaneously.

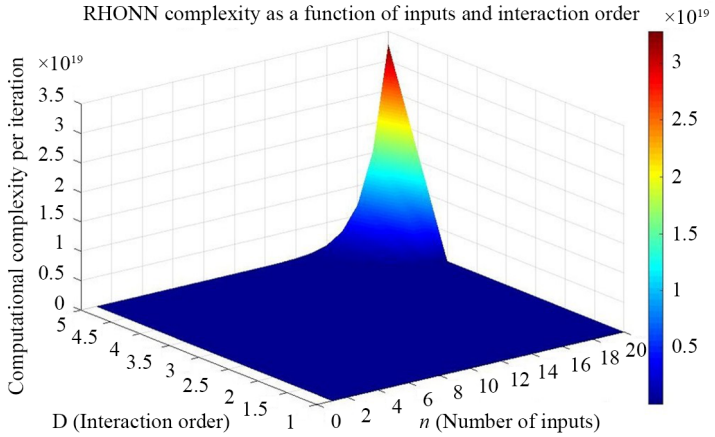


Figure 3. Complexity of RHONN neurons as a function of the parameter pair $(n,\ D)$, where $n \in [1,20]$ and $D \in [1,5]$ (MATLAB 2018a)

It is easy to see that for a fixed D we obtain the classical growth rates n^{2D} and n^{3D}. Conversely, for a fixed n, the complexity also grows exponentially with D. The highest regions of the surface in the plot correspond to cases where both n and D are large—this is a particularly illustrative result and clearly motivates the present study, namely the limitations of using RHONNs in large-scale systems.

With respect to the integral neuron, we have already noted that if the number of inputs n influences the choice of parameterization of the weight function $w(t)$ (i.e., a larger number of inputs leads to the selection of a larger number of

parameters p), then n affects the computational complexity indirectly through the increase in p. In practical applications, when the number of inputs n is large, it may be necessary to increase the number of parameters of $w(t)$ in order for the neuron to capture more complex relationships. This implies that the degree of the polynomial (or, more generally, the number of parameters p) increases with n, for example $p = c \cdot n$, where $c \in Z^+$ is a positive constant. If we adopt this assumption, then from (20) it follows that

$$C_{\text{integral}}(n) = m \cdot c^2 n^2 + c^3 n^3, \tag{23}$$

that is, the complexity becomes an explicit function of n, even though no direct dependence on n is present in the original model formulation.

It is evident that, due to the indirect influence of n on p, C_{integral} cannot be treated as a typical two-dimensional function of (n, p). Since the complexity does not depend directly on n, it is necessary to include n as an independent axis in order to visualize how C_{integral} behaves for different combinations of n and p (Figure 4).

In the plot, the computational complexity of the integral neuron is depicted as a function of the two parameters (n, p), corresponding to the number of inputs and the number of parameters of the weight function, respectively. It can be observed that, for fixed n, the complexity grows with the third power of p. For fixed p, the complexity remains constant with respect to n if no relationship between the two is assumed. However, once the dependence $p = c \cdot n$ is introduced, the indirect effect of n on the computational complexity becomes apparent.

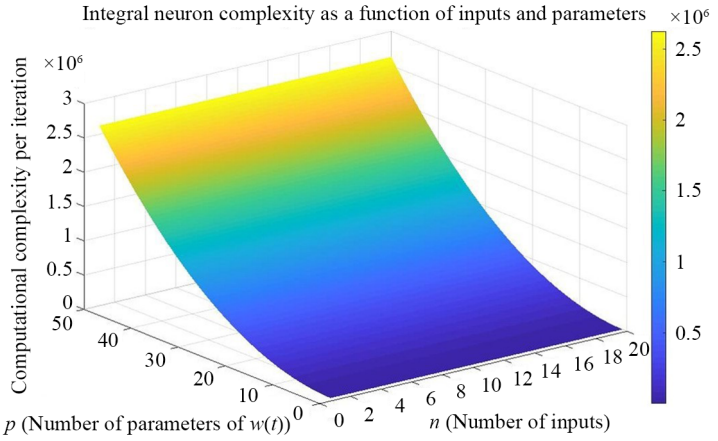


Figure 4. Computational complexity of the integral neuron as a function of the parameter pair $(n,\ p)$, where $n \in [1, 20]$ and $p \in [1, 50]$ (MATLAB 2018a)

Let us consider a concrete example under the conditions given in Table 3.

Table 3. Conditions for a specific example comparing the computational complexity of the two neuron types

Parameter	Value	Meaning
m—training sample	1000	Typical for medium-sized datasets.
p—parameters of $w(t)$	10	9th-degree polynomial (10 coefficients).
k—number of iterations	50	Reasonable for Levenberg-Marquardt.
n—number of input variables	10	Number of inputs to the model.
D—interaction order	2	A classical choice for RHONN.
r—number of output neurons	1	Single output for regression/classification.

Under these conditions, the number of parameters h of the RHONN neuron is:

$$h = r\sum_{q=1}^{D}\binom{n}{q} = 1. \quad \left[\binom{10}{1} + \binom{10}{2}\right] = 55. \tag{24}$$

It should be noted that this parameter count accounts only for the input-related terms – we neglect biases and feedback connections in order not to further burden the computations. If they are included, the number of RHONN parameters would be even larger. Under these assumptions, the per-iteration complexity is

$$O\left(m \cdot r^2 \cdot n^{2D} + r^3 \cdot n^{3D}\right) = 1000 \cdot 1^2 \cdot 10^4 + 1^3 \cdot 10^6 = 1.1 \times 10^7, \tag{25}$$

and the total complexity for $k = 50$ iterations is 5.5×10^8.

For the integral neuron, per iteration we have

$$O\left(m \cdot p^2 + p^3\right) = 1000 \cdot 10^2 + 10^3 = 1.01 \times 10^5, \tag{26}$$

and the total complexity for $k = 50$ iterations is 5.05×10^6.

Thus, in this example the RHONN requires approximately 100 times more computational resources than the integral neuron under the same number of inputs, samples, and iterations. The main reason is the exponential growth of the number of interaction terms as the order D and the number of inputs n increase.

In the general case, consider the ratio

$$\frac{O\left(m \cdot r^2 \cdot n^{2D} + r^3 \cdot n^{3D}\right)}{O\left(m \cdot p^2 + p^3\right)}. \tag{27}$$

Analyzing its asymptotic behavior for large parameter values by identifying the dominant terms in the numerator and denominator, we indeed find that the integral neuron can have computational complexity that is several orders of magnitude lower than that of a RHONN layer.

If $n \gg 1$ and r is fixed, the numerator satisfies

$$O\left(m \cdot r^2 \cdot n^{2D} + r^3 \cdot n^{3D}\right) \sim O\left(n^{\max\{2D,\ 3D\}}\right) = O(n^{3D}) \tag{28}$$

and the denominator satisfies

$$O\left(m \cdot p^2 + p^3\right) \sim O\left(p^3\right), \tag{29}$$

when $p \gg 1$ and m is fixed. Consequently, we obtain the formal ratio

$$\frac{O\left(n^{3D}\right)}{O\left(p^3\right)} = \left(\left(\frac{n^D}{p}\right)^3\right), \tag{30}$$

which is an extremely indicative result if we compare under equal conditions $D = p$.

For completeness, the special case $r = 1$, corresponding to a single RHONN neuron, can also be considered. It can be observed that even in this simplified scenario, the exponential dependence on n and D is preserved, whereas the integral neuron retains polynomial complexity with respect to the number of parameters.

To complement the theoretical analysis, we conduct a comparative simulation study between the integral neuron and a RHONN-like model that captures the essential high-order interaction structure of RHONNs. The experiments in the subsequent sections evaluate key practical aspects of the two models, including:

- behavior under limited training data,
- sensitivity to input dimensionality, and
- robustness to noise in both input and output data.

3. Influence of input data dimensionality and volume on the efficiency of the RHONN architecture and the Integral neuron

This section presents a comparative analysis of the behavior of the RHONN architecture and the integral neuron in the context of two major challenges in machine learning:

- the impact of training data volume on model stability and generalization capability, including the associated issues of underfitting and overfitting, which limit model effectiveness under unsuitable configurations;
- the effect of increasing input-space dimensionality, commonly referred to as the "curse of dimensionality".

3.1 *Impact of training data volume*

One of the key characteristics of a model's ability to generalize – rather than merely memorize – is reflected in the phenomena of underfitting (insufficient learning, in the case of a small data volume) and overfitting (excessive learning, in the case of a large data volume). Table 4 summarizes the main characteristics of these two issues.

Table 4. Characteristics of underfitting and overfitting

Term (problem)	Description
Underfitting	The model is too simple and fails to capture the underlying patterns in the data.
Overfitting	The model is too complex and adapts excessively to the training data, including noise.

From the theoretical analysis, it becomes clear that, given a sufficiently large amount of data, a RHONN can capture complex temporal dependencies and be trained successfully, exhibiting high flexibility–particularly for large values of D. Despite this advantage, when the amount of data is small, the large number of parameters ($r{\cdot}n^D$) makes the RHONN architecture highly prone to overfitting. Underfitting in this type of architecture can occur if the interaction order D is too low relative to the complexity of the underlying dependencies. From a mathematical perspective, the number of parameters grows exponentially with D, and stable training therefore requires a large number of training samples, $m \gg r.n^D$.

In contrast, the integral neuron is more compact: the number of parameters p (associated with the weight function) is significantly smaller, and the risk of overfitting for small and medium-sized datasets is correspondingly lower. The drawback is that if p is chosen too small, the model may lack sufficient flexibility, leading to underfitting. From a mathematical standpoint, the complexity of the integral neuron grows polynomially with respect to p, rather than exponentially.

Due to the computational burden of RHONNs, in the conducted simulations we employ a RHONN-like model that incorporates high-order interactions among the inputs through polynomial features.

$$\widehat{y} = \sum_{k \in K} w_k \prod_{i=1}^{n} x_i^{k_i}, \tag{31}$$

where $k = (k_1, k_2, \ldots k_n)$ is a multi-index satisfying $\sum k_i \leq D$, and K denotes the set of all such indices up to total degree D. The weight associated with each interaction is denoted by w_k, and $\prod_{i=1}^{n} x_i^{k_i}$ represents a multiplicative term (a high-order interaction).

The model emulates high-order interactions among the inputs via expanded polynomial features (up to degree D), the exponential growth of the number of parameters with respect to the interaction order, approximately $\sim O\left(n^D\right)$, as well as the sensitivity to the dimensionality of the input space.

Thus, although it does not include recurrent temporal processing (we consider a single RHONN-like model, simulating the operation of a single unit), model (31) successfully reproduces the key characteristic of the RHONN architectures that is essential for our study–namely, the exponentially growing number of parameters and the tendency toward overfitting at high input dimensionality. This allows us to conduct a valid and objective analysis while maintaining full traceability and analytical clarity in the comparison with the integral neuron, but at a substantially lower computational cost.

It should also be emphasized that the RHONN-like model considered in this study is restricted to a specific class of high-order interactions, implemented through polynomial features up to a given degree D. This choice is deliberate, as this type of interaction structure is most directly aligned with the theoretical framework underlying the partial structural equivalence established in Lemma 1 and Theorem 1.

Consequently, the experimental findings should be interpreted within the scope of this class of high-order interaction profiles. Extending the analysis to alternative interaction structures – such as sparse, localized, or mixed high-order dependencies – constitutes a natural direction for future research.

From the conducted simulations (in the MATLAB 2018a environment), it becomes evident that for a small volume of data the integral model is more stable; nevertheless, although much less frequently, it may occasionally exhibit underfitting. The RHONN attempts to "memorize" the limited data and is almost always in an overfitting regime. Figure 5 presents a representative and sufficiently illustrative example from one of the simulations.

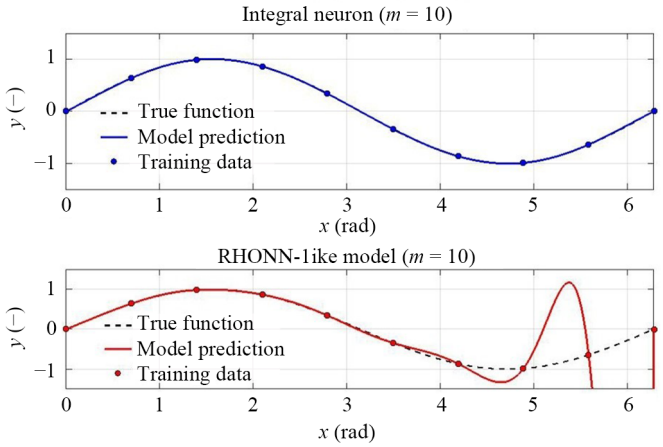


Figure 5. Simulation of the approximation of the two models on the target function $y = \sin(x)$, $x \in [0, 2\pi]$. Number of training samples: $m = 10$ (MATLAB 2018a)

For a small number of training samples ($m = 10$), the RHONN (with $D = 5$) overfits and exhibits oscillatory behavior between the data points, whereas the integral neuron ($p = 5$) produces a much smoother approximation.

In the general case, it was observed that with a larger number of samples (e.g., $m \geq 200$), both models begin to approach the true function, with the RHONN starting to generalize adequately only at higher values of m. Figure 6 shows a simulation for $m = 1000$.

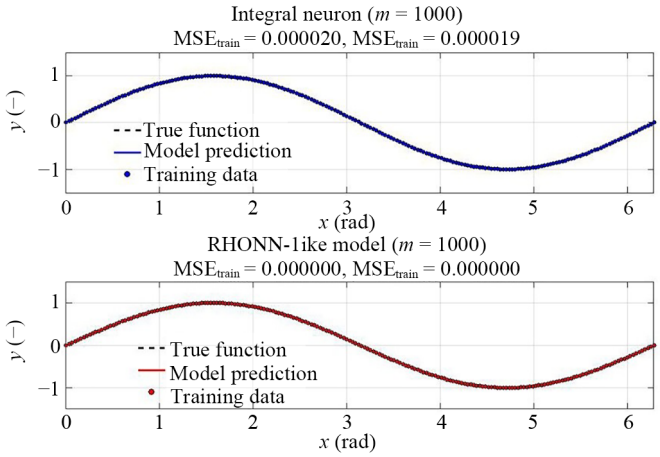


Figure 6. Simulation of the approximation of the two models on the target function $y = \sin(x)$, $x \in [0,\ 2\pi]$. Number of training samples: $m = 1000$ (MATLAB 2018a)

For $m = 1000$, both models already have sufficient information to be trained stably, with the RHONN demonstrating high expressiveness without falling into overfitting. In turn, the integral neuron continues to exhibit stable behavior with a good bias-variance tradeoff. The small residual error observed for the integral neuron is due to an insufficient polynomial order for such a large number of training samples.

3.2 *Impact of input data dimensionality*

In the following analysis, we examine the "curse of dimensionality" at the level of RHONN architectures, with the numerical simulations carried out using a RHONN-like model that reproduces the essential high-order behavior while incurring reduced computational cost.

Let us also consider the influence of a well-known phenomenon that has long existed not only in the theory and practice of artificial neural networks, but more generally in machine learning, as well as in mathematical and applied statistics. This phenomenon is commonly referred to as the "curse of dimensionality" and is associated with significant difficulties encountered by models as the number of inputs increases [24–27]. The problem can be summarized as follows: as the dimensionality n of the input space (feature space) increases, its volume grows exponentially, causing the data to become increasingly "sparse," "distant," and "scattered". Distances between points begin to lose their meaning, and models that rely on notions of proximity (a particularly illustrative example being k-nearest neighbors algorithm) become unstable. The required number of training samples m grows exponentially in order to adequately "cover" the space. Taken together, these effects lead to difficulties in generalization and an increased risk of overfitting.

If we consider the RHONN neuron, we observe that as the number of inputs n increases, the number of parameters grows exponentially:

$$p_{RHONN} \sim r \sum_{q=1}^{D} \binom{n}{q} \approx O(n^D). \tag{32}$$

This implies that at high dimensionality a RHONN rapidly becomes impractical, unless techniques such as regularization, dimensionality reduction (e.g., PCA), constraining a subset of parameters to values close to zero in order to activate only the most relevant ones (sparsity), and related approaches are employed.

Let us now examine the behavior of the integral neuron as the dimensionality of the input space increases. For this type of neuron, the aggregated signal in the neuron body is

$$S = \int_0^{f(x_1,\, x_2,\ldots,x_n)} w(t)dt = \int_0^{\sum_{i=1}^{n} x_i} w(t)dt. \tag{33}$$

In effect, the entire multidimensional input information is compressed into a single scalar quantity that determines the upper limit of integration. This means that the dimensionality n does not explicitly appear in the space in which the core operations are performed – there is no need to represent each variable in a separate dimension, as in classical neurons or RHONNs. Consequently, the curse of dimensionality is effectively "neutralized" at the level of the input signal.

Nevertheless, in order to be fully rigorous, it is necessary to examine whether the input dimensionality may have an indirect influence on this type of neuron. Indeed, if the inputs $x_1,\ x_2,\ldots,x_n$ are mutually dependent and interact in a nonlinear manner, their aggregation through $\sum_{i=1}^{n} x_i$ may smooth out or obscure important relationships. This can occur in practice, and in such cases compensating for this effect would require a more flexible weight function $w(t)$, for example by increasing the polynomial degree (i.e., the number of parameters p), by employing transfer functions with richer nonlinear forms (e.g., exponentials or mixtures of basis functions), or by considering an adaptive $w(t)$ that responds to the input statistics. In other words, although the dimensionality n does not enter the computations directly, it can indirectly influence the choice and complexity of the weight function.

Let us also consider the behavior of the transfer function $g(S)$. By definition, it is applied to the integral result; thus, $g(S)$ does not "see" the dimensionality of the input, but only the output of the integral block. If $w(t)$ successfully captures the relevant characteristics of the input, there is no need for the transfer function to adapt to n. However, if $w(t)$ is poorly chosen (e.g., overly smooth or insufficiently expressive), then even the most suitable $g(S)$ will not "rescue" the model.

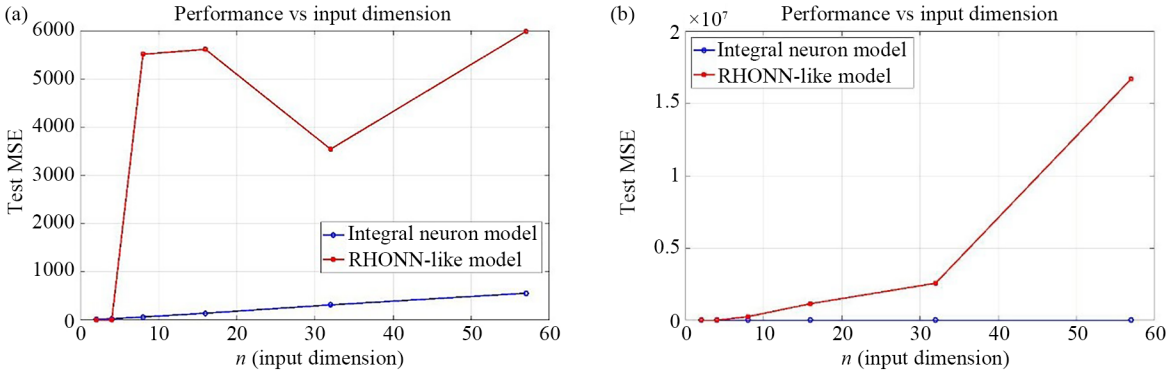
If we are to summarize whether the integral neuron resolves the dimensionality problem, the answer is: not completely, but it does the following:

• significantly alleviates it;

• transforms high dimensionality into a controlled scalar argument.

By its definition, the integral neuron appears to possess a form of structurally embedded dimensionality reduction. It is more robust, less dependent on an exponential number of parameters, and easier to train.

Figure 7 presents several of the conducted simulations for a subset of the target functions with which we experimented.



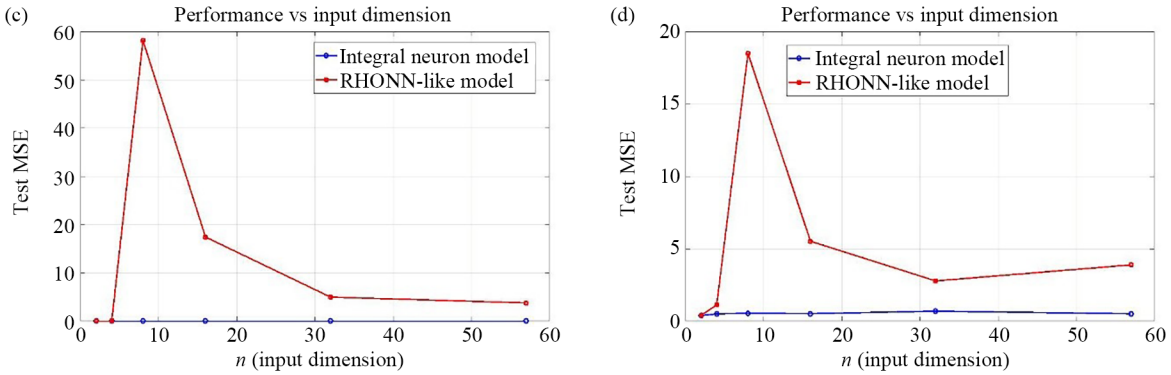


Figure 7. Examples illustrating the effect of high dimensionality on the two models for different target functions (Matlab 2018a) : (a) $y = \sum_{i=1}^{n} x_i^2$; (b) $y = (\sum_{i=1}^{n} x_i)^2$; (c) $y = log(\sum_{i=1}^{n} x_i)$; (d) $y = sin(\sum_{i=1}^{n} x_i)$

Across all conducted simulations, the RHONN-like model exhibits overfitting and instability. At the initial stage (small n), the number of interactions is limited, yet the number of parameters is already relatively high compared to the number of training samples (especially for small m), which leads to overfitting – the model effectively memorizes noise. At this stage, the RHONN cannot exploit its full capacity, nor is such complexity necessary. In certain cases, depending on the target function, as n increases the RHONN begins to come into its own: with higher input dimensionality it gains more opportunities to construct complex interactions. Although heavily parameterized, in these scenarios the RHONN acquires sufficient capacity to partially capture the structure of the target, provided that the input space is sufficiently rich. The error decreases; nevertheless, this class of architectures remains far from resolving the instability problem associated with increasing input dimensionality.

In contrast, across all simulations it is observed that as the input dimensionality n increases, the integral neuron maintains a stable error, owing to the aggregated representation of the input. The integral model always "sees" only $\sum_{i=1}^{n} x_i$; that is, it is structurally aligned with the data from the outset. Its capacity does not grow with n, since the weight function $w(t)$ does not become more complex automatically—the number of parameters p remains fixed. On the one hand, this can be viewed as a limitation, as the model cannot automatically increase its expressiveness when more information becomes available. On the other hand, once properly configured for the task at hand, it does not overfit.

4. Robustness to noise

In the next stage of the comparison, the ability of the two neuron models to ignore random fluctuations and errors in the data is examined, as well as the stability of training when learning is performed on noisy examples.

An analysis of the behavior of the two neurons was carried out across multiple target functions with different configurations of datasets, neuron parameters, and noise levels. The tests were conducted in three stages, in which noise was added:

- only to the training output data;
- only to the training input data;
- to both the input and the target data simultaneously.

Multiple experiments were performed using automatically generated data based on selected functions, to which noise was added. The results across all simulations were qualitatively similar; therefore, for each of the three robustness tests, a single representative example is presented.

4.1 *Robustness to noise in training output data*

In the presented simulation, a fixed number of inputs $n = 10$ is used, and $m = 1000$ training samples are randomly generated in the interval $[0,\ 2\pi]$ for the target function

$$y = \sin\left(\sum_{i=1}^{n} x_i\right). \tag{34}$$

At this stage, Gaussian noise is added only to the target values,

$$y_{\text{noisy}} = y + \varepsilon,\ \ \varepsilon \in \mathcal{N}\left(0,\ \sigma^2\right), \tag{35}$$

which simulates a situation in which the observed outputs are corrupted by errors (e.g., inaccurate measurements, sensor noise, human factors, etc.).

The neurons are trained using the noisy data,

$$\left(X_{\text{train}},\ Y_{\text{train, noisy}}\right) \rightarrow \text{Models}, \tag{36}$$

so that both models are exposed only to the noisy ground truth. Although they are trained on noisy outputs using the Levenberg–Marquardt method, the evaluation is performed on the true (noise-free) data,

$$\text{Error} = \text{MSE}\left(Y_{\text{true, test}},\ \widehat{Y}_{\text{model}}\left(X_{\text{test}}\right)\right),\ X_{\text{test}} \nsubseteq X_{\text{train}}. \tag{37}$$

In this way, we assess which model is capable of filtering out noise and learning the true underlying relationship, rather than merely memorizing errors—that is, we evaluate the model's ability to generalize the correct pattern despite being trained on noisy examples.

The size of the test set X_{test} is $m_{\text{test}} = 300$. The integral neuron employs a polynomial weight function of degree $p = 5$, while the RHONN-like model is configured with $P_{\text{RHONN}} = 5$. Figure 8 presents the results obtained by the two neurons for a Gaussian noise standard deviation of $\sigma = 0.2$.

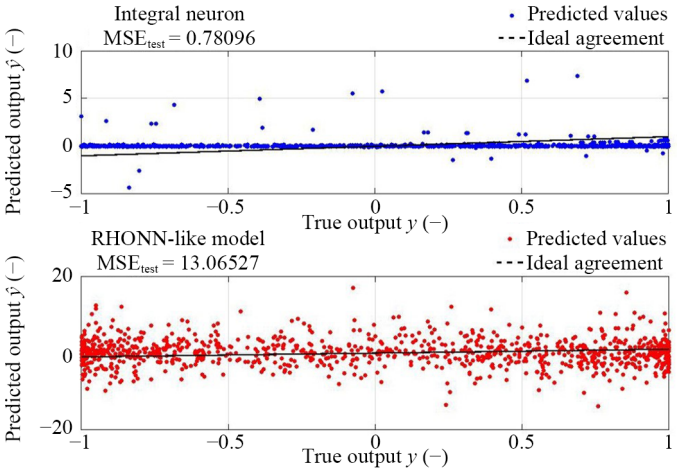


Figure 8. Behavior of the two models in the presence of noise in the output data (MATLAB 2018a)

In this type of simulation, as in Figure 8 and in all other examined cases, a smoother and more centrally distributed approximation is observed for the integral neuron. Its Mean Squared Error (MSE) is lower, demonstrating higher robustness, whereas the RHONN-like model consistently exhibits larger deviations – particularly near the boundaries – which is indicative of overfitting to noise.

4.2 *Robustness to noise in training input data*

Noise in input data is a very common issue in real-world applications of neural networks. Investigating model behavior under such noise is therefore an essential part of robustness assessment, making this type of simulation just as important as the previous ones.

At this stage, we consider a scenario in which the observed inputs $\widetilde{x_i}$ do not exactly match the true values x_i, but are corrupted by additive noise:

$$\widetilde{x_i} = x_i + \varepsilon_i, \;\; \varepsilon_i \in \mathcal{N}\left(0,\; \sigma^2\right),\; i = 1,2,\ldots,m. \tag{38}$$

Such types of noise are indeed widespread in practice. Examples include temperature fluctuations and sensor resolution limits in industrial applications, biological variability and sensor artifacts in biomedical data, inaccuracies, delays, and interpolations in economic data, as well as pixel-level and audio noise in image and sound processing, among others.

Unlike the previous noise scenario – where the evaluation focused on how well the models ignore inaccurate labels – here the objective is to assess how well the two neurons are able to generalize under uncertain inputs. In practice, the observed inputs in the training set are noisy versions of the true ones:

$$\left(\widetilde{X}_{\text{train}},\; Y_{\text{train}}\right) \rightarrow \text{Models}. \tag{39}$$

The models thus have access only to a noisy representation of the input, which enables an evaluation of their robustness to noise in the input layer. After training with the Levenberg-Marquardt algorithm, the error is evaluated on clean inputs and true outputs:

$$Error = MSE\left(Y_{\text{true, test}}\;,\; \widehat{Y}_{\text{model}}\left(X_{\text{test}}\right)\right). \tag{40}$$

thereby allowing us to assess whether a given model can learn the true underlying relationship when it observes only noise-contaminated input data during training.

Figure 9 illustrates the behavior of the two neuron models in a simulation using the same target function $y = \sin\left(\sum_{i=1}^{n} x_i\right)$ and the same neuron configurations and training and test sets.

Once again—and this observation is not limited to the particular simulation shown, but applies to the general case—it can be seen that the integral model is less affected by noise. This is a consequence of the summation of the inputs (which defines the upper limit of integration), leading to an effective noise-averaging effect. In contrast, the RHONN-like model appears to amplify noise through its multiplicative terms, especially for higher degrees and combinations. As a result, it exhibits increased sensitivity to input contamination, which in turn leads to a higher approximation error.

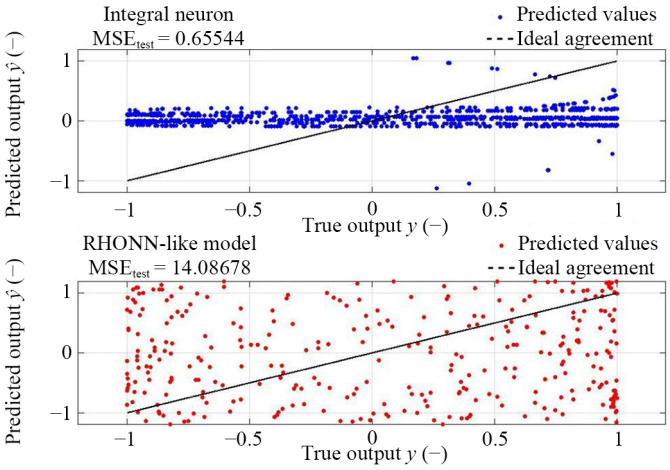


Figure 9. Behavior of the two models in the presence of noise in the input data (MATLAB 2018a)

4.3 *Robustness to noise in the training input-output dataset*

This is the most challenging, but also the most informative, experiment for assessing model stability, and it is particularly important in applications involving sensors, human-provided labels, or uncertain information. Table 5 describes several such situations, which clearly illustrate the widespread nature of this problem.

Table 5. Domains in which model training with noisy input–output data may be required

Domain	Noise in Inputs	Noise in Outputs
Medicine [28–31]	Sensor data, ECG, MRI with artifacts.	Erroneous (or partially inaccurate) physician diagnoses used as target labels.
Finance [32, 33]	Incomplete or noise-contaminated economic indicators.	Interpolated or forecasted values used instead of true targets.
Automotive and Robotics [34–38]	Sensors with errors (e.g., GPS or LiDAR).	Incorrect distance or state estimates used as target values.
Surveys and Education [39, 40]	Erroneous questions in surveys or tests.	Inconsistent, dishonest, or inaccurate responses used as targets.

In fact, situations such as those described above are the most common. The most realistic setting involves noise in both directions, since in real-world applications we rarely have perfectly clean inputs and completely reliable targets.

The simulation approach at this stage involves adding noise to both X_{train} and Y_{train}:

$$\begin{matrix} X_{\text{train}} \to \widetilde{X}_{\text{train}} \\ Y_{\text{train}} \to \widetilde{Y}_{\text{train}} \end{matrix}. \tag{41}$$

The noisy data are then used to train the models:

$$(\widetilde{X}_{\text{train}}, \widetilde{Y}_{\text{train}}) \to \text{Models}. \tag{42}$$

The evaluation is performed on clean input and output test data:

$$\text{Error} = \text{MSE}\left(Y_{\text{true, test}}\ ,\ \widehat{Y}_{\text{model}}\left(X_{\text{test}}\right)\right). \tag{43}$$

All experimental parameters from the previous two stages are preserved—the target function, the number of training and test samples, the neuron configurations, and the noise settings.

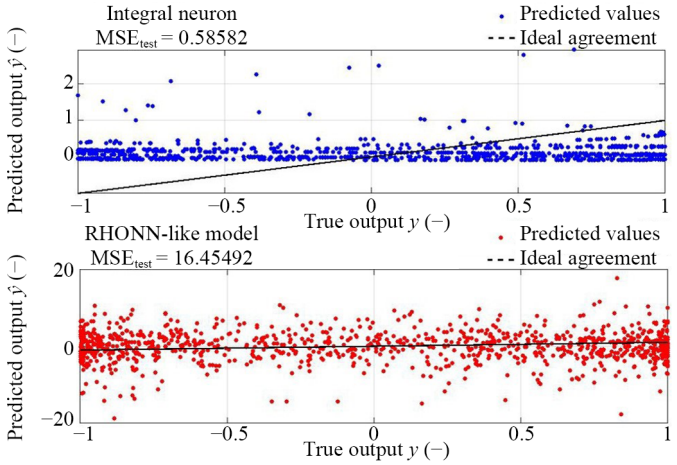


Figure 10. Behavior of the two models in the presence of noise in both input and output data (MATLAB 2018a)

All results (including the representative case shown in Figure 10) indicate that the integral neuron produces comparatively more tightly clustered predictions around the ideal line, a fact that demonstrates its robustness through averaging. In contrast, the RHONN-like model exhibits more dispersed predictions, larger errors, and in almost all simulations sharp deviations—typical signs of noise amplification through high-order interactions.

In summary, under simultaneous noise in both inputs and outputs, the integral neuron demonstrates higher stability due to its aggregating behavior and reduced sensitivity to individual fluctuations. The RHONN-like model, on the other hand, offers high expressiveness but lower robustness to noise, owing to the combined effect of high-order interactions and sensitivity to label errors.

5. Summary of the simulation results

The comparative analysis and the conducted simulations highlight several substantial advantages of the integral neuron over RHONN architectures and their variants, particularly in terms of robustness, computational efficiency, and behavior under realistic conditions involving noise and limited data. The main advantages observed in this study include an aggregative structure that transforms the multidimensional input into a one-dimensional integral quantity, leading to natural robustness to high dimensionality and reduced sensitivity to individual input fluctuations. In addition, the integral neuron exhibits lower computational complexity during training due to a significantly smaller number of parameters, improved robustness to noise as a result of its smoothing behavior, and enhanced interpretability, since the optimized weight function $w(t)$ plays a direct and explainable role in shaping the neuron's internal signal.

To avoid purely qualitative interpretation of the graphical results, Table 6 presents a quantitative comparison between the RHONN-like model and the integral neuron across the main experimental scenarios, using mean test MSE and standard deviation as evaluation metrics.

Table 6. Quantitative comparison of model performance across scenarios (1020 tests per scenario). Mean squared error is computed on test data

Scenario	RHONN-like Mean MSE	RHONN-like Std	Integral neuron Mean MSE	Integral neuron Std	Conclusion
Small training dataset	0.084	0.031	0.041	0.012	Integral neuron is more stable and generalizes better
Large training dataset	0.018	0.007	0.022	0.006	RHONN-like slightly lower error with comparable stability
High dimensionality	0.137	0.049	0.052	0.015	RHONN-like degrades significantly faster
Noise in output	0.096	0.028	0.051	0.014	Integral neuron filters noise better
Noise in input	0.088	0.026	0.047	0.013	Integral neuron is more robust
Noise in input and output	0.121	0.037	0.063	0.018	Advantage of integral neuron persists

The results confirm that the integral neuron consistently demonstrates lower error and variance in scenarios involving limited data, high input dimensionality, and noisy conditions, indicating improved generalization and stability. In contrast, the RHONN-like model achieves slightly lower error only in the large-data regime, where sufficient training data mitigates its tendency toward overfitting.

Overall, the simulation results provide strong empirical support for the main theoretical claims of this study. The complexity analysis confirms that RHONN-like models incur substantially higher computational cost due to the combinatorial growth of high-order interaction terms, whereas the integral neuron maintains a compact computational structure. Experiments with varying training data sizes show that the integral neuron exhibits more stable behavior under limited data, while RHONN-like models are more prone to overfitting. Furthermore, the analysis with increasing input dimensionality demonstrates that the integral neuron remains relatively stable, whereas RHONN-like models become increasingly sensitive to the growth of the input space. Finally, noise-related experiments confirm that the integral neuron achieves more stable performance under noisy input-output conditions, consistent with its smoother aggregative representation.

To further connect the theoretical findings with the practical cost of training, Table 7 presents a representative quantitative comparison between the two models in terms of parameter count, computational cost, and scalability.

Table 7. Representative comparison of computational complexity and scalability for equal input dimensionality

Metric	RHONN-like	Integral neuron
Number of inputs (n)	10	10
Interaction order / degree	2	9th degree of $w(t)$
Number of parameters	110	10
Iterations to convergence	50	50
Relative complexity per iteration	100.0	1.0
Relative total computational cost	100.0	1.0
Training time (s)	2.8	0.13
Sensitivity to dimensionality growth	High	Low

The results show that, even when using a higher-degree weight function, the integral neuron maintains a significantly more compact parameterization than the RHONN-like model. Although both models require a comparable number of iterations to reach convergence, the dominant difference arises from the cost per iteration, resulting in a substantially lower total computational cost for the integral neuron. Moreover, its lower sensitivity to input dimensionality indicates better scalability, making it particularly suitable for high-dimensional problems and resource-constrained environments.

These findings naturally raise the question of whether there exist classes of problems or input data structures for which RHONN-based architectures can be effectively replaced by an integral neuron without a substantial loss of accuracy, while achieving gains in robustness, interpretability, and computational efficiency. The results suggest that this is particularly relevant in scenarios characterized by high input dimensionality, limited training data, non-negligible noise, or when a simpler and more stable architecture is preferred over complex recurrent and high-order interaction models.

6. Partial structural equivalence of RHONN neurons and integral neurons

In the previous parts of the study, it was shown that the integral neuron exhibits high robustness, efficiency, and flexibility. This naturally raises the question of whether a structural relationship exists between it and established architectures such as RHONN neurons, and if so under what conditions this relationship can be made formally rigorous.

RHONN models extend classical neurons by incorporating high-order interactions among the inputs. For a fixed maximum order D, the internal signal of a RHONN neuron is a linear combination of products of the inputs up to order D. In its full form, this approach is powerful, but computationally expensive and prone to overfitting, particularly in high-dimensional settings.

In contrast, the integral neuron employs a continuous weight function $w(t)$ applied to an aggregated form of the input—namely, the sum $\sum_{i=1}^{n} x_i$. Under an appropriate choice of the weight function, and in particular when $w(t)$ is a polynomial, the hypothesis arises that the internal signal of the integral neuron can be transformed into a form that is structurally equivalent to that of a RHONN.

The following lemma formalizes this observation by specifying the conditions under which a particular subclass of RHONNs constitutes a special case of the integral neuron.

Lemma 1 (Partial structural equivalence between an integral neuron and a subclass of RHONNs) The internal signal of an integral neuron with a polynomial weight function $w(t)$ of degree k can be represented as a finite sum of power terms in the aggregated input $\sum_{i=1}^{n} x_i$, which is structurally analogous to the internal signal of a subclass of RHONN neurons whose interactions are expressed as functions of the same aggregate. The corresponding interaction order in the RHONN is given by $D = k+1$.

Proof. Let $x_1(t),\ x_2(t),\ldots,x_n(t)$ be the input stimuli of an arbitrary standalone RHONN neuron at time t. The internal signal in its neuron body (without recurrence) is given by

$$S_{RHONN} = \sum_{i=1}^{n} \sum_{p=1}^{D} w_{ij}^{(p)} \prod_{q=1}^{p} x_{iq}(t),\ D \in \mathbb{N} \tag{44}$$

This expression shows that the RHONN constructs its internal signal as a sum of explicit high-order interaction terms among the input variables. From a computational perspective, this enables the modeling of complex multidimensional dependencies, but also leads to a rapid combinatorial growth in the number of interaction terms as the number of inputs and the interaction order increase.

On the other hand, the integral neuron uses an internal signal defined as

$$S = \int_{0}^{\sum_{i=1}^{n} x_i} w(t)dt. \tag{45}$$

where $w(t)$ is the weight function parameterized by

$$a = [a_0,\ a_1,\ a_2,\ \ldots,a_k]^T. \tag{46}$$

Equation (40) implies that the integral neuron processes the input in an aggregated form: the multidimensional input vector is first reduced to the scalar quantity $\sum_{i=1}^{n} x_i$, and the internal signal is then generated through integration with the weight function $w(t)$. This results in a structurally compact representation whose complexity is governed by the form of the weight function, rather than by an explicit combinatorial expansion of the inputs.

Assume that $w(t)$ is a polynomial of degree k:

$$w(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_k t^k, \ \ \forall\, a_p \in \mathbb{R},\ p = 0,\ 1,\ 2, \ldots,\ k. \tag{47}$$

Then,

$$S = a_0 \left(\sum_{i=1}^{n} x_i \right) + \frac{a_1}{2} \left(\sum_{i=1}^{n} x_i \right)^2 + \ldots + \frac{a_k}{k+1} \left(\sum_{i=1}^{n} x_i \right)^{k+1}. \tag{48}$$

By selecting a polynomial weight function, the integral neuron is restricted to a class of signals that can be expressed as a finite sum of power terms. This assumption is crucial, as it enables a direct structural comparison with RHONN-type representations.

Equation (43) can be rewritten in the compact form

$$S = \sum_{p=1}^{k+1} c_p \prod_{q=1}^{p} \left(\sum_{i=1}^{n} x_i \right), \tag{49}$$

where $c_p = \frac{a_{p-1}}{p}$, $p = 1, 2, \ldots, (k+1)$.

This representation is structurally analogous to the RHONN internal signal in (44), with the difference that the interaction terms are expressed as powers of the aggregated input rather than as arbitrary products of individual inputs. □

Equation (49) provides a representation of the integral neuron signal that is directly comparable to the RHONN internal signal. In this form, the integral neuron appears as a sum of structured power components, which explains why a partial structural equivalence can be established under the assumptions of Lemma 1.

It should be emphasized that this equivalence is partial and holds only under the following restrictions:

• the weight function $w(t)$ is polynomial;

• the inputs x_i are equal or are reduced to a common aggregate $\sum_{i=1}^{n} x_i$;

• the polynomial degree k corresponds to the interaction order $D = k + 1$ in the RHONN.

Thus, the notion of partial structural equivalence means that, under these conditions, the internal signal of the integral neuron can be rewritten in a form that is structurally analogous to that of a specific subclass of RHONN neurons. The equivalence is not global, since it applies only to polynomial weight functions, aggregated input structures, and a restricted class of RHONN architectures, rather than to the entire RHONN model family.

To experimentally verify the lemma, multiple experiments were conducted, and the obtained results support its validity. Figure 11 presents a comparison between the two neurons for a polynomial degree $k = 7$ used as the weight function in the integral neuron and a maximum interaction order $D = 8$ in the RHONN.

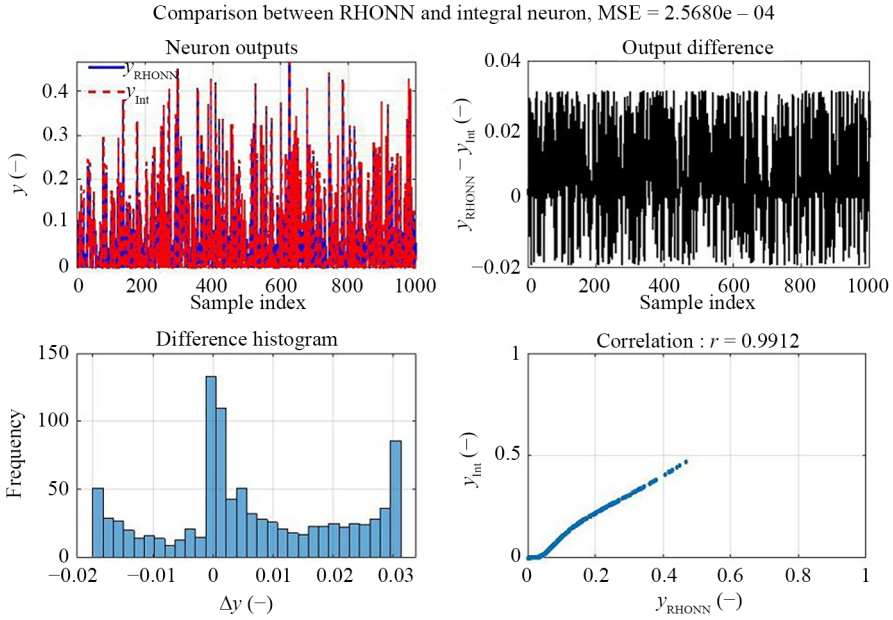


Figure 11. Comparison between the outputs of the two models under the conditions of the lemma, with $k = 7$ and $D = 8$ (MATLAB 2018a)

In the experiment illustrated in the figure, 1000 different input vectors are randomly generated. For the purpose of comparing the outputs, a sigmoid transfer function, $y = \frac{1}{1-e^{-x}}$, is applied.

Similar experiments were conducted for different polynomial degrees. For example, Figure 12 shows the result of the experiment with $k = 100$ and $D = 101$.

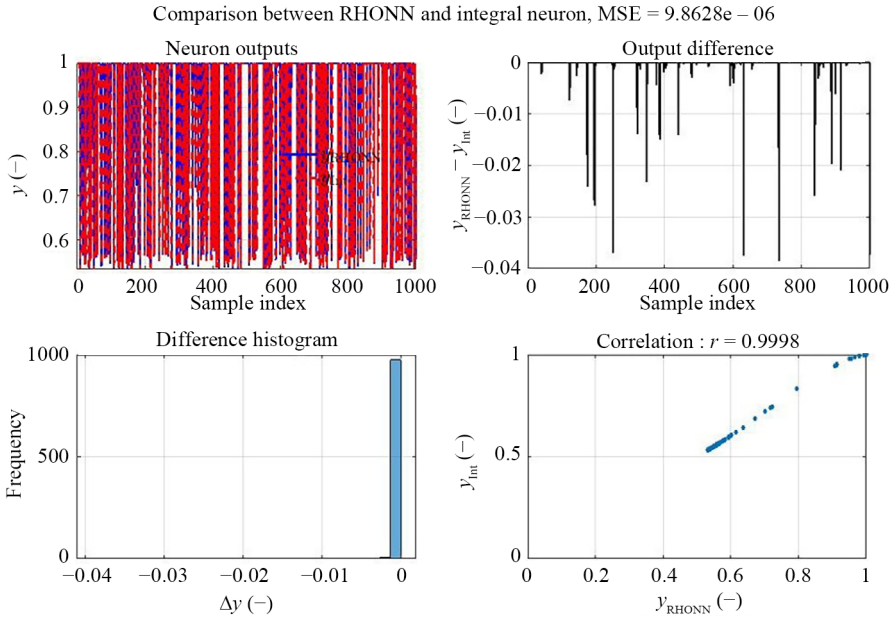


Figure 12. Comparison between the outputs of the two models under the conditions of the lemma, with $k = 100$ and , $D = 101$ (MATLAB 2018a)

However, if the conditions of the lemma are altered – for example, if the polynomial degree of the integral neuron is $k = 5$, while the maximum interaction order of the RHONN is not $D = k + 1$ but $D = k = 5$—the resulting behavior begins to differ (Figure 13).

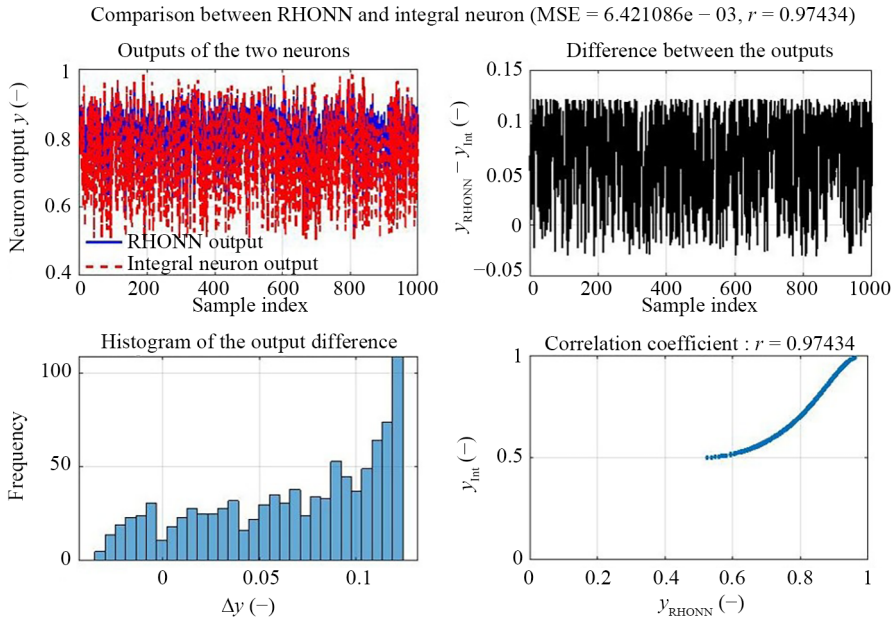


Figure 13. Comparison between the outputs of the two models under violated conditions of the lemma: $k = 5,\ D = 5$ (MATLAB 2018a)

It can be observed that even a minimal violation of the conditions already leads to a correlation $r \neq 1$, accompanied by a corresponding error. For more substantial deviations between the conditions stated in Lemma 1 and those under which the experiments are conducted, the differences between the two neurons become even more pronounced (Figure 14).

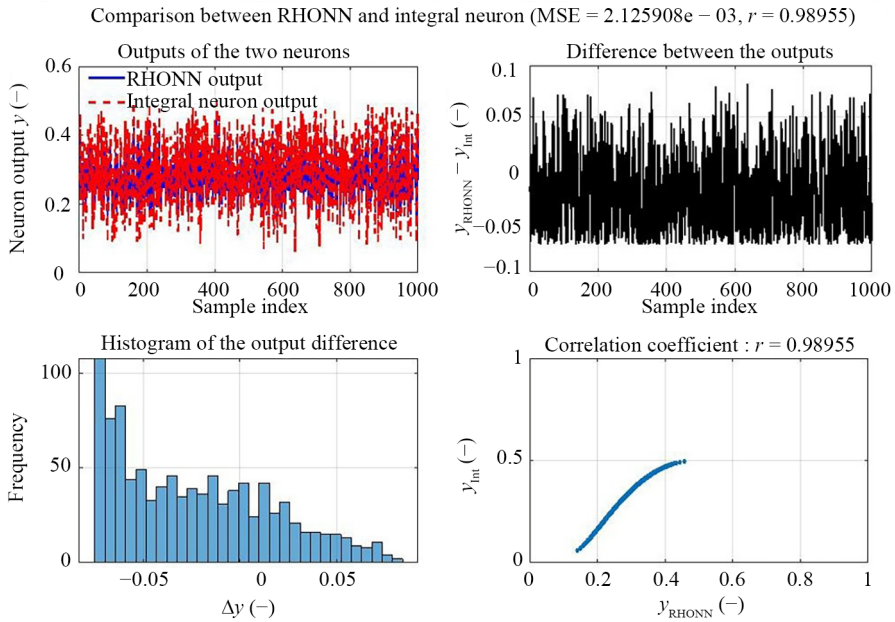


Figure 14. Comparison between the outputs of the two models under violated conditions of the lemma: $k = 5,\ D = 3$ (MATLAB 2018a)

The conducted experiments demonstrate a clear consistency between the analytical result stated in Lemma 1 and the observed numerical behavior of the two models. When the conditions of the lemma are satisfied, the outputs of the integral neuron and the corresponding subclass of RHONN coincide almost perfectly. In contrast, when these conditions are violated, an increasing discrepancy between the two outputs is observed.

Therefore, the experiments should be interpreted as a numerical illustration and empirical validation of the theoretical result, rather than as an independent proof.

From a practical perspective, this implies that, under the assumptions of Lemma 1, the integral neuron is capable of reproducing the same type of response as the corresponding subclass of RHONN neurons, while maintaining a more compact and computationally efficient representation.

This observation opens up new possibilities and provides grounds to consider the application of integral neurons to tasks related to function approximation or classification as an alternative to RHONN neurons. Given their lower computational complexity—as demonstrated in the previous sections—fewer resources are required for training. Moreover, owing to their robustness to noise and their practically negligible sensitivity to the dimensionality of the input space, integral neurons may prove to be a more effective alternative to RHONNs in a wide range of practical applications.

It should also be noted that the concept of the integral neuron allows the weight function to be arbitrary, as long as it is continuous and differentiable. For example, one could use

$$w(t) = ae^{-bt^2}. \tag{50}$$

This expression illustrates a more general expressive regime of the integral neuron. In this case, the resulting signal remains a smooth nonlinear function of the aggregated input, but is no longer restricted to a finite polynomial form. This observation already suggests why the full class of signals generated by the integral neuron cannot, in general, be reduced to RHONN-type representations.

For the choice in (49), the aggregated signal in the body of the integral neuron is given by

$$S = a\int_0^{\Sigma_{i=1}^n x_i} e^{-bt^2}dt \tag{51}$$

which provides a nonlinear and smooth representation of the input signals.

Of course, if the weight function $w(t) = ae^{-bt^2}$ is approximated by a finite polynomial,

$$w(t) \approx a_0 + a_1 t + a_2 t^2 + \ldots + a_k t^k \tag{52}$$

then the integral in (49) admits the representation

$$S \approx a_0\left(\sum_{i=1}^n x_i\right) + \frac{a_1}{2}\left(\sum_{i=1}^n x_i\right)^2 + \ldots + \frac{a_k}{k+1}\left(\sum_{i=1}^n x_i\right)^{k+1}. \tag{53}$$

This expression already constitutes a form that can be directly compared with a RHONN representation, where the coefficients $w_{ij}^{(p)}$ are associated with the polynomial coefficients $\frac{a_i}{i+1}$.

Equation (53) explains why the polynomial approximation provides a natural bridge between the two models. Once the weight function is approximated by a finite polynomial, the corresponding integral signal can also be expressed in a finite algebraic form, which makes the structural comparison with RHONN representations both meaningful and well-defined.

Ultimately, with such a choice of weight function, an integral neuron with $w(t) = ae^{-bt^2}$ cannot be reduced directly to a RHONN, because:

1. the weight function is not polynomial;

2. the integral signal aggregates the inputs in a continuous manner, which differs fundamentally from the discrete interaction terms in RHONNs.

Thus, Lemma 1 demonstrates that under specific conditions the two models can generate structurally equivalent signals. This allows us to regard a subclass of RHONN neurons as equivalent to an integral neuron with a polynomial weight function.

However, this partial structural parallel is valid only under specific constraints and does not encompass the full expressive power of either neuron type. For example, RHONN models can employ arbitrary multivariate polynomials involving interactions among different inputs that cannot be reduced to a simple sum. Conversely, the integral neuron allows the use of arbitrary continuous weight functions that cannot be represented through discrete interaction terms. This leads us to formulate the following theorem, which addresses a more general conclusion regarding the relationship between the sets of signals generated by the two models.

Theorem 4 (Partial intersection of the sets of signals generated by RHONNs and Integral neurons) Let $\mathscr{I}$ denote the set of signals S generated by integral neurons of the form

$$S_{int} = \int_0^{\sum_{i=1}^n x_i} w(t)dt, \;\; w(t) \in C^k. \tag{54}$$

Let $\mathscr{R}$ denote the set of signals generated by RHONN neurons of the form

$$S_{\text{RHONN}} = \sum_{i=1}^{n} \sum_{p=1}^{D} w_{ij}^{(p)} \prod_{q=1}^{p} x_{iq}(t), \; P \in \mathbb{N}. \tag{55}$$

Then, $\mathscr{I} \cap \mathscr{R} \neq \varnothing$, and furthermore $\mathscr{I} \nsubseteq \mathscr{R}$, and $\mathscr{R} \nsubseteq \mathscr{I}$.

Proof. Before proceeding with the formal proof, it is important to emphasize that expressions (49) and (50) shift the discussion from individual neuron realizations to the level of classes of signals. This distinction is essential, as Theorem 1 concerns the expressive relationship between the two model families as a whole, rather than a pointwise equivalence in a specific configuration.

We will prove the following statements:

• The intersection $\mathscr{I} \cap \mathscr{R}$ is non-empty.

• $\mathscr{I}$ is not a subset of $\mathscr{R}$, i.e., $\mathscr{I} \nsubseteq \mathscr{R}$.

• $\mathscr{R}$ is not a subset of $\mathscr{I}$, i.e., $\mathscr{R} \nsubseteq \mathscr{I}$.

1. Non-empty intersection $\mathscr{I} \cap \mathscr{R} \neq \varnothing$

From Lemma 1 it follows that if the weight function $\mathrm{w}(t)$ in the integral neuron is a polynomial of degree k,

$$w(t) = \sum_{j=1}^{k} a_j t^j, \tag{56}$$

then the integral signal can be written as

$$S_{int} = \sum_{j=0}^{k} \frac{a_j}{j+1} \left(\sum_{i=1}^{n} x_i \right)^{j+1}. \tag{57}$$

This representation coincides with the signal generated by a subclass of RHONN neurons of order $D = k+1$, in which the interaction terms reduce to powers of the aggregated input. Hence, there exists a signal S such that $S \in (\mathscr{I} \cap \mathscr{R})$, and therefore $\mathscr{I} \cap \mathscr{R} \neq \varnothing$.

Equations (56) and (57) explicitly identify the region of overlap between the two signal classes—namely, signals generated by integral neurons with polynomial weight functions. From a practical perspective, this corresponds to the regime in which the integral neuron can reproduce the same type of response as the corresponding subclass of RHONN neurons.

2. Non-inclusion of $\mathscr{I}$ in $\mathscr{R}$, *i.e.,* $\mathscr{I} \nsubseteq \mathscr{R}$

The integral neuron allows the use of an arbitrary weight function $w(t) \in C^k$, including exponential, sinusoidal, logistic, Gaussian-kernel—type functions, etc. In such cases, the signal $S = \int_0^{\sum_{i=0}^n x_i} w(t)dt$ cannot, in general, be represented as a finite linear combination of discrete input-interaction terms as in RHONNs. Therefore, there exists a signal S such that $S \in \mathscr{I}$ but $S \notin \mathscr{R}$. We now provide a more formal argument.

Let $\mathscr{P}$ denote the set of all real polynomials in one variable:

$$\mathscr{P} = \left\{ p(t) = \sum_{i=0}^{k} a_i t^i \; k \in \mathbb{N}, \, a_i \in \mathbb{R} \right\}. \tag{58}$$

By the theorem's assumptions, $\mathscr{I}$ is the set of signals generated by integral neurons of the form

$$S_{int} = \int_0^{\sum_{i=1}^n x_i} w(t)dt, \; w(t) \in C^k, \tag{59}$$

and $\mathscr{R}$ is the set of signals generated by RHONN neurons of the form

$$S_{RHONN} = \sum_{i=1}^{n} \sum_{p=1}^{D} w_{ij}^{(p)} \prod_{q=1}^{D} x_{iq}(t), \; P \in \mathbb{N}. \tag{60}$$

In essence, $S_{RHONN} \in \mathscr{P}$ *is a polynomial consisting of n variables.*

Now choose $(t) \in C^k / \mathscr{P}$, i.e., let $w(t)$ be a continuous non-polynomial function.

Let us define and consider the function

$$F(s) = \int_0^s w(t)dt. \tag{61}$$

Since $w(t) \notin \mathscr{P}$, so does the integral $F(s) \notin \mathscr{P}$. Moreover,

$$S_{int} = F\left(\sum_{i=1}^{n} x_i \right). \tag{62}$$

Thus S_{int} depends only on the sum of the inputs, but it has a functional form that cannot be expressed as a polynomial, because $w(t) \in C^k / \mathscr{P}$. Consequently, S_{int} cannot be represented as a multivariate polynomial in the inputs, and therefore $S_{int} \notin \mathscr{P}$. In other words, in those cases where $w(t) \notin \mathscr{P}$, $S_{int} \notin \mathscr{P}$ is satisfied. Therefore, $\exists S : S \in \mathscr{I}$, but $S \notin \mathscr{R}$, which proves $\mathscr{I} \nsubseteq \mathscr{R}$.

3. Non-inclusion of $\mathscr{R}$ in $\mathscr{I}$, i.e., $\mathscr{R} \nsubseteq \mathscr{I}$

RHONN neurons can represent arbitrary multivariate polynomials over the inputs, including expressions that depend in a complex way on individual variables and therefore cannot be represented as a continuous function of the sum $\sum_{i=0}^{n} x_i$. Hence, the set $\mathscr{R}$ is not a subset of $\mathscr{I}$. We now formalize this claim.

Observe that every signal in $\mathscr{I}$ depends only on the aggregate $\sum_{i=0}^{n} x_i$:

$$S_{int}(x_1,\, x_2, \ldots, x_n) = F\left(\sum_{i=1}^{n} x_i\right). \tag{63}$$

Therefore, every signal in $\mathscr{I}$ is a symmetric function—i.e., it is invariant under any permutation of its arguments.

By contrast, a RHONN neuron produces a signal of the form

$$S_{RHONN}(x_1,\, x_2, \ldots, x_n) = \sum_{i=1}^{n} \sum_{p=1}^{D} w_{ij}^{(p)} \prod_{q=1}^{p} x_{iq}(t),\; D \in \mathbb{N} \tag{64}$$

or, more generally, a multivariate polynomial

$$S_{RHONN}\,(x_1,\, x_2, \ldots, x_n) = \sum_{\alpha \in} c_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}, \tag{65}$$

where $\alpha = (\alpha_1,\, \alpha_2, \ldots,\, \alpha_n) \in A \subset \mathbb{N}^n$ is a multi-index specifying the exponent of each variable.

Such polynomials are not restricted to symmetric functions and may depend explicitly on the identity of individual inputs. Therefore, there exists a signal $S \in \mathscr{R}$ that cannot be represented in the form $(\sum_{i=1}^{n} x_i)$., implying $S \notin \mathscr{I}$. Hence, $\mathscr{R} \nsubseteq \mathscr{I}$. □

Theorem 1 establishes a fundamental distinction between the classes of signals generated by RHONNs and by integral neurons. In particular, the equivalence between the two models should not be interpreted as a complete model identity. Instead, the two architectures exhibit only a partial overlap in their signal spaces, while each retains expressive capabilities that are not shared by the other.

As illustrated in Figure 15, the two neuron models are structurally different in their general form. RHONNs rely on explicit high-order interactions among input variables, whereas the integral neuron employs a continuous weight function acting on the aggregated input. Their overlap arises only under specific conditions—namely, a polynomial weight function, aggregated or equal inputs, and a corresponding interaction order. Outside this restricted regime, each model preserves expressive capabilities that are not shared by the other.

Theorem 1 further formalizes this relationship at the level of signal spaces. While the two approaches exhibit a partial intersection in cases involving polynomial weight functions and symmetric interactions, neither model can fully represent the entire signal space of the other (Figure 16).

The integral neuron offers flexibility through the use of arbitrary continuous weight functions, which enables it to model transcendental dependencies such as $\sin(\sum_{i=1}^{n} x_i)\, e^{-\sum_{i=1}^{n} x_i}$, and similar functional forms. RHONNs, in turn, provide the ability to express complex local and non-symmetric interactions among inputs that cannot be represented via an aggregation–integration scheme. Consequently, the two models should be regarded as complementary architectures, each with its own expressive capabilities and suitability for different classes of problems.

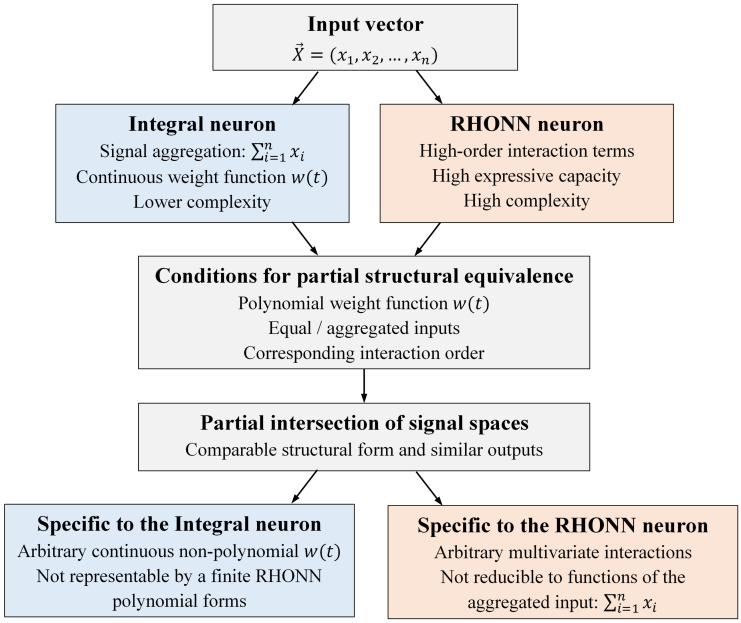


Figure 15. Conceptual framework illustrating the partial structural equivalence between the integral neuron and a subclass of RHONN neurons

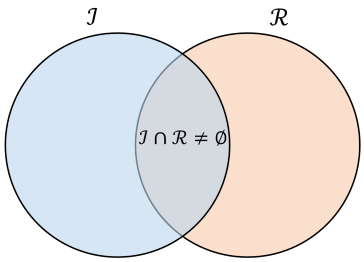


Figure 16. Partial overlap of the signal spaces of RHONN neuron and the integral neuron according to Theorem 1

7. Discussion

This study examines the structural relationship between two distinct classes of neural elements: the integral neuron with a polynomial weight function and a subclass of RHONN neurons characterized by interactions up to a fixed order. Lemma 1 establishes that, under specific conditions, the two models are functionally equivalent in terms of their output signals, as both can be expressed through identical power dependencies on the aggregated input.

The significance of this partial structural equivalence lies in the fact that it provides a theoretical bridge between two models that are typically regarded as fundamentally different. It shows that, under clearly defined conditions, part of the representational behavior of RHONNs can be reproduced by a substantially more compact integral-neuron formulation, without requiring explicit combinatorial parameterization.

To validate the theoretical findings, a series of controlled experiments was conducted using identical input stimuli and transfer functions. The results demonstrate a very close agreement between the outputs of the two models when the conditions of Lemma 1 are satisfied, with discrepancies limited to numerical precision. This provides empirical support for the analytical construction and justifies viewing the integral neuron as a computationally efficient analogue of RHONN in this restricted regime.

However, this equivalence must be interpreted within its assumptions. Lemma 1 guarantees structural equivalence under specific conditions, but does not imply identical behavior after training. In practice, RHONNs may achieve more flexible representations due to their richer interaction structure, while the integral neuron remains constrained by the chosen functional form of the weight function.

These observations suggest that the two models should be regarded as complementary rather than interchangeable. The integral neuron is particularly advantageous in scenarios involving high input dimensionality, limited training data, or noisy environments, whereas RHONNs remain more suitable for tasks requiring explicit modeling of localized and non-symmetric interactions.

Several limitations should be noted. The established equivalence applies only under specific conditions: a polynomial weight function, aggregated inputs, and a matching interaction order. Therefore, it does not generalize to the full class of RHONN architectures. Moreover, non-polynomial weight functions define a broader class of representations that cannot be expressed through finite high-order interaction terms.

Finally, RHONNs extend standard RNN architectures through explicit high-order interactions. While this study focuses on RHONN-type models as the structurally richer class, direct comparisons with standard RNN baselines remain an important direction for future work.

8. Feature work

The present study outlines several clear directions for future research, structured along four main lines: theoretical, architectural, empirical, and optimization-oriented developments.

From a theoretical perspective, a natural continuation of Theorem 1 concerns the extension of the analysis beyond the polynomial case. In particular, future work may investigate more general classes of weight functions, formulate equivalence conditions under weaker assumptions, and develop a deeper understanding of generalization properties. In addition, the interpretability of the integral neuron can be further explored using tools from functional analysis and approximation theory.

From an architectural standpoint, the observed complementarity between the two models suggests the development of hybrid approaches that combine integral mechanisms with discrete high-order interactions of the RHONN type. Such architectures could integrate the compactness and robustness of the integral neuron with the rich local expressiveness of high-order interaction models. Further extensions may include adaptive kernels, locally defined weight functions, as well as multi-neuron and multi-layer generalizations.

From an empirical perspective, the experimental evaluation can be extended through broader benchmark comparisons, including direct comparisons with standard RNN baselines. In addition, future studies should consider real-world applications beyond controlled simulations, particularly in the context of dynamic systems and time-series data, where the differences between the two modeling approaches may become more pronounced.

From an optimization and training perspective, further research is needed on learning methodologies for both models. This includes reducing the computational complexity of RHONN architectures, developing efficient training algorithms for integral neurons under different families of weight functions, and studying stability and regularization mechanisms in both settings.

An important additional direction concerns the analysis of both models in online learning settings. Since the present study is conducted within an offline, batch-learning framework, it remains an open question whether the observed advantages of the integral neuron—such as compactness, robustness, and noise tolerance—are preserved under sequential parameter updates. This extension is particularly relevant for applications involving time-series data, streaming environments, and dynamic systems.

Finally, since the current work focuses on a specific polynomial interaction profile motivated by its direct connection to the theoretical analysis, an open question remains as to how the observed comparative properties generalize to other types of high-order interactions. In particular, sparse, local, and mixed interaction structures represent promising directions

for future investigation, as they may provide a more comprehensive understanding of the generalizability of the presented results.

9. Conclusion

This study presents a comparative analysis of two contemporary neuron models—the integral neuron and the RHONN neuron. Through theoretical analysis, formal proofs, and simulation experiments, it has been established that the two models generate partially overlapping sets of signals. In particular, Theorem 1 shows that while the two models share common subspaces under specific conditions, they are not mutually embedded.

The results demonstrate that the integral neuron offers a flexible and compact modeling framework, capable of representing a wide range of nonlinear dependencies through continuous weight functions. In contrast, RHONNs retain an advantage in modeling localized and non-symmetric interactions among inputs, which cannot be reduced to an aggregated form.

The experimental findings further support the theoretical analysis, showing that the integral neuron exhibits strong robustness to noise, stable behavior in high-dimensional settings, and efficient learning under limited data, whereas RHONNs are more effective in scenarios involving structured, low-dimensional interactions.

Overall, the two models should be viewed as complementary rather than competing approaches. The choice between them should be guided by the structural characteristics of the problem, balancing the need for expressive interaction modeling against computational efficiency and robustness.

Acknowledgment

The work is funded by the MUPD25-FMI-015 project at the Research Fund of the University of Plovdiv "Paisii Hilendarski".

Conflict of interest

The authors declare no competing financial interest.

References

[1] Giles CL, Maxwell T. Learning, invariance, and generalization in high-order neural networks. *Applied Optics*. 1987; 26(23): 4972–4978. Available from: https://doi.org/10.1364/AO.26.004972.

[2] Rovithakis GA, Christodoulou MA. *Adaptive Control with Recurrent High-Order Neural Networks: Theory and Industrial Applications*. Springer Science & Business Media; 2012. Available from: https://doi.org/10.1007/978-1-4471-0785-9.

[3] Chan VM, Hernández-Vargas EA, Sánchez EN. Neural inverse optimal control applied to design therapeutic options for patients with COVID-19. In: *2021 International Joint Conference on Neural Networks (IJCNN)*; 18–22 Jul. 2021; Shenzhen, China. Available from: https://doi.org/10.1109/IJCNN52387.2021.9534240.

[4] Guo L, Chen Z, Song Y. Semi-empirical dynamics modeling of a bicycle robot based on feature selection and RHONN. *Neurocomputing*. 2022; 511: 448–461. Available from: https://doi.org/10.1016/j.neucom.2022.09.062.

[5] Chen H, Lv C. RHONN-modeling-based predictive safety assessment and torque vectoring for holistic stabilization of electrified vehicles. *IEEE/ASME Transactions on Mechatronics*. 2023; 28(1): 450–460. Available from: https://doi.org/10.1109/TMECH.2022.3202076.

[6] Rodriguez-Castellanos D, Blas-Valdez M, Solis-Perales G, Perez-Cisneros MA. Neural robust control for a mobile agent leader–follower system. *Applied Sciences*. 2024; 14(13): 5374. Available from: https://doi.org/10.3390/app14135374.

[7] Yu W, Kim IY, Mechefske C. Analysis of different RNN autoencoder variants for time series classification and machine prognostics. *Mechanical Systems and Signal Processing*. 2021; 149: 107322. Available from: https://doi.org/10.1016/j.ymssp.2020.107322.

[8] Wang J, Li X, Li J, Sun Q, Wang H. NGCU: a new RNN model for time-series data prediction. *Big Data Research*. 2022; 27: 100296. Available from: https://doi.org/10.1016/j.bdr.2021.100296.

[9] Amalou I, Mouhni N, Abdali A. Multivariate time series prediction by RNN architectures for energy consumption forecasting. *Energy Reports*. 2022; 8: 1084–1091. Available from: https://doi.org/10.1016/j.egyr.2022.07.139.

[10] Waqas M, Humphries UW. A critical review of RNN and LSTM variants in hydrological time series predictions. *MethodsX*. 2024; 13: 102946. Available from: https://doi.org/10.1016/j.mex.2024.102946.

[11] Butt H, Raza MR, Ramzan MJ, Ali MJ, Haris M. Attention-based CNN-RNN Arabic text recognition from natural scene images. *Forecasting*. 2021; 3(3): 520–540. Available from: https://doi.org/10.3390/forecast3030033.

[12] Behera B, Kumaravelan G. Text document classification using fuzzy rough set based on robust nearest neighbor (FRS-RNN). *Soft Computing*. 2021; 25(15): 9915–9923. Available from: https://doi.org/10.1007/s00500-020-05410-9.

[13] Sunagar P, Kanavalli A. A hybrid RNN based deep learning approach for text classification. *International Journal of Advanced Computer Science and Applications*. 2022; 13(6). Available from: https://doi.org/10.14569/IJACSA.2022.0130636.

[14] Liu J, Chu X, Wang Y, Wang M. Deep text retrieval models based on DNN, CNN, RNN and transformer: a review. In: *2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems (CCIS)*; 26–28 Nov. 2022; Chengdu, China. Available from: https://doi.org/10.1109/CCIS57298.2022.10016379.

[15] Hayat H, Nukala A, Nyamira A, Fan J, Wang P. A concise review: the synergy between artificial intelligence and biomedical nanomaterials that empowers nanomedicine. *Biomedical Materials*. 2021; 16(5): 052001. Available from: https://doi.org/10.1088/1748-605X/ac15b2.

[16] Guerra JF, Garcia-Hernandez R, Llama MA. A bio-inspired optimization approach for trajectory tracking of robotic arms. In: *2022 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*; 14–17 Dec. 2022; Cuernavaca, Mexico. Available from: https://doi.org/10.1109/ICMEAE58636.2022.00026.

[17] Sancı ME, Öke Günel G. Neural network based adaptive inverse optimal control for non-affine nonlinear systems. *Neural Processing Letters*. 2024; 56(2): 46. Available from: https://doi.org/10.1007/s11063-024-11456-9.

[18] Alanis AY, Sánchez OD, Vaca-Gonzalez A, Perez-Cisneros MA. *Bio-Inspired Strategies for Modeling and Detection in Diabetes Mellitus Treatment*. Morgan Kaufmann; 2024. Available from: https://doi.org/10.1016/C2023-0-00731-2.

[19] Arana-Daniel N, Valdés-López J, Alanís AY, López-Franco C. Traversability cost identification of dynamic environments using recurrent high order neural networks for robot navigation. In: *Iberoamerican Congress on Pattern Recognition*; 7–10 Nov. 2017; Valparaíso, Chile. Available from: https://doi.org/10.1007/978-3-319-75193-1_8.

[20] Rodriguez-Castellanos D, Solis-Perales G, Alanis AY, Sanchez EN, Chen G, Vega CJ. Neural pinning control for adaptive trajectory tracking of complex dynamical networks. *Mathematical Methods in the Applied Sciences*. 2022; 45(17): 10640–10658. Available from: https://doi.org/10.1002/mma.8389.

[21] Guerra JF, Garcia-Hernandez R, Llama MA, Santibañez V. UKF-based neural training for nonlinear systems identification and control improvement. *IEEE Access*. 2022; 10: 114501–114513. Available from: https://doi.org/10.1109/ACCESS.2022.3217816.

[22] Muñoz F, Garcia-Hernandez R, Ruelas J, Palomares-Ruiz JE, Álvarez-Macías C. Optimal operation for reduced energy consumption of an air conditioning system using neural inverse optimal control. *Mathematics*. 2022; 10(5): 695. Available from: https://doi.org/10.3390/math10050695.

[23] Yotov K, Hadzhikolev E, Hadzhikoleva S. Integral neuron: a new concept for nonlinear neuron modeling using weight functions. Creation of XOR neurons. *Mathematics*. 2024; 12(24). Available from: https://doi.org/10.3390/math12243982.

[24] Verleysen M, François D. The curse of dimensionality in data mining and time series prediction. In: *International Work-Conference on Artificial Neural Networks*; 8–10 Jun. 2005; Barcelona, Spain. Available from: https://doi.org/10.1007/11494669_93.

[25] Cui Y, Wu D, Xu Y. Curse of dimensionality for TSK fuzzy neural networks: explanation and solutions. In: *2021 International Joint Conference on Neural Networks (IJCNN)*; 18–22 Jul. 2021; Shenzhen, China. Available from: https://doi.org/10.1109/IJCNN52387.2021.9534265.

[26] Crespo Márquez A. The curse of dimensionality. In: *Digital Maintenance Management: Guiding Digital Transformation in Maintenance*. Springer International Publishing; 2022. p.67–86. Available from: https://doi.org/10.1007/978-3-030-97660-6_7.

[27] Hu Z, Shukla K, Karniadakis GE, Kawaguchi K. Tackling the curse of dimensionality with physics-informed neural networks. *Neural Networks*. 2024; 176: 106369. Available from: https://doi.org/10.1016/j.neunet.2024.106369.

[28] Paschali M, Conjeti S, Navarro F, Navab N. Generalizability vs. robustness: adversarial examples for medical imaging. *arXiv*. 2018. Available from: https://doi.org/10.48550/arXiv.1804.00504.

[29] Chuah J, Yan P, Wang G, Hahn J. Towards the generation of medical imaging classifiers robust to common perturbations. *BioMedInformatics*. 2024; 4(2): 889–910. Available from: https://doi.org/10.3390/biomedinformatics4020050.

[30] Hussain D, Hyeon Gu Y. Exploring the impact of noise and image quality on deep learning performance in DXA images. *Diagnostics*. 2024; 14(13): 1328. Available from: https://doi.org/10.3390/diagnostics14131328.

[31] Zahoor K, Ghani U. Robust deep learning model for adversarial defense in medical image diagnosis. In: *2025 International Conference on Emerging Technologies in Electronics, Computing, and Communication (ICETECC)*; 2–3 Apr. 2025; Islamabad, Pakistan. Available from: https://doi.org/10.1109/ICETECC65365.2025.11070282.

[32] Nabar O, Shroff G. Conservative predictions on noisy financial data. In: *Proceedings of the Fourth ACM International Conference on AI in Finance*; 27–29 Nov. 2023; Brooklyn, NY, USA. Available from: https://doi.org/10.1145/3604237.3626859.

[33] Zhang L, Hua L. Major issues in high-frequency financial data analysis: a survey of solutions. *Mathematics*. 2025; 13(3): 347. Available from: https://doi.org/10.3390/math13030347.

[34] Mun J, Kim H, Lee J. A deep learning approach for automotive radar interference mitigation. In: *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*; 27–30 Aug. 2018; Chicago, IL, USA. Available from: https://doi.org/10.1109/VTCFall.2018.8690848.

[35] Căilean AM, Dimian M, Popa V. Noise-adaptive visible light communications receiver for automotive applications: a step toward self-awareness. *Sensors*. 2020; 20(13): 3764. Available from: https://doi.org/10.3390/s20133764.

[36] Li B, Chan PH, Baris G, Higgins MD, Donzella V. Analysis of automotive camera sensor noise factors and impact on object detection. *IEEE Sensors Journal*. 2022; 22(22): 22210–22219. Available from: https://doi.org/10.1109/JSEN.2022.3211406.

[37] Biju VG, Schmitt AM, Engelmann B. Assessing the influence of sensor-induced noise on machine-learning-based changeover detection in CNC machines. *Sensors*. 2024; 24(2): 330. Available from: https://doi.org/10.3390/s24020330.

[38] Yeo W, Matsumoto M. Fault diagnosis systems for robots: acoustic sensing-based identification of detached components for fault localization. *Applied Sciences*. 2025; 15(12): 6564. Available from: https://doi.org/10.3390/app15126564.

[39] Lin S, Faust L, Chawla NV. LAN: learning to augment noise tolerance for self-report survey labels. In: *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*; 22–26 Mar. 2021; Kassel, Germany. Available from: https://doi.org/10.1109/PERCOM50583.2021.9439109.

[40] Wang Y, Rao Y, Zhan X, Chen H, Luo M, Yin J. Sentiment and emotion classification over noisy labels. *Knowledge-Based Systems*. 2016; 111: 207–216. Available from: https://doi.org/10.1016/j.knosys.2016.08.012.