Knowledge-Based Systems 265 (2023) 110379

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Neural Logic Circuits: An evolutionary neural architecture that can learn and generalize

Hamit Taner Ünal [a,*], Fatih Başçiftçi [b]

[a] Selçuk University, Institute of Natural and Applied Sciences, Department of Information Technologies Engineering, Konya, Turkey
[b] Selçuk University, Faculty of Technology, Department of Computer Engineering, Konya, Turkey

ARTICLE INFO

Article history:
Received 21 June 2022
Received in revised form 4 February 2023
Accepted 7 February 2023
Available online 10 February 2023

Keywords:
Neuroplasticity
Neural Networks
Artificial Intelligence
Logic Gates
Genetic Algorithms

ABSTRACT

We introduce Neural Logic Circuits (NLC), an evolutionary, weightless, and learnable neural architecture loosely inspired by the *neuroplasticity* of the brain. This new paradigm achieves learning by evolution of its architecture through reorganization of augmenting synaptic connections and generation of artificial neurons functioning as logic gates. These neural units mimic biological nerve cells stimulated by binary input signals and emit excitatory or inhibitory pulses, thus executing the "*all-or-none*" character of their natural counterparts. Unlike Artificial Neural Networks (ANN), our model achieves generalization ability without intensive weight training and dedicates computational resources solely to building network architecture with optimal connectivity. We evaluated our model on well-known binary classification datasets using advanced performance metrics and compared results with modern and competitive machine learning algorithms. Extensive experimental data reveal remarkable superiority of our initial model, called NLCv1, on all test instances, achieving outstanding results for implementation of this new paradigm.

"As long as our brain is a mystery, the universe, the reflection of the structure of the brain will also be a mystery."

[Santiago Ramón y Cajal, 1903]

1. Introduction

The human brain is one of the most advanced and complicated systems ever existed. For centuries, a great deal of effort has been devoted to understanding its working mechanism and imitating its remarkable capabilities with the aim to design machines that can think, learn, understand, and make decisions like humans. The advent of digital computers and advances in the field of Artificial Intelligence (AI) have all been inspired by the biological facts provided by research in neuroscience.

The human brain accommodates a massive network of biological nerve cells called ***neurons.*** Neurons communicate with each other through a link between them called ***synapses***. The exchange of information between neurons is carried out via chemicals named neurotransmitters. A typical neuron can have thousands of synaptic connections with other neurons. Research in the field of neuroscience reveals that our nervous system operates using an electrochemical process, and each neuron fires in discrete pulses with an ***all-or-none*** character. Neurons take multiple inputs from other neurons and produce a digital output, either ***excite*** or ***inhibit*** the incoming signals.

Throughout our lifetime, the neural circuitry of our brain is subject to functional and structural reorganization. It can restructure neural pathways, create new connections, and prune existing ones – a process called ***Neuroplasticity*** [1]. The brain can also form new neurons and eliminate unused ones, commonly known as *Neurogenesis*. These structural modifications are considered as the basis of *learning* and the ability of the brain to adapt to changes in the environment.

Brain neuroplasticity is learning-driven, meaning frequently used synapses are strengthened while seldom-used ones are weakened or pruned. It is also known as the "*use it or lose it rule*". The strength of synaptic connections depends on the intensity or repetition of stimuli or learning activity. While neurons can temporarily enhance connections by releasing more neurotransmitters, reinforcement of activity or sustainment of internal or external stimuli can produce structural changes in the neural network, such as the growth of dendritic spines, establishment of new connections, and generation of new neurons. This explains the learning process in terms of short-term and long-term memory. The complexity of the neural circuitry of our brain's cerebral cortex is often considered to be a measure of intelligence [2].

Starting in the 1930s, several approaches have been proposed to characterize and mimic the behavior of nerve cells [3–7]. With

* Corresponding author.
E-mail address: htaner.unal@gmail.com (H.T. Ünal).

https://doi.org/10.1016/j.knosys.2023.110379
0950-7051/

the development of symbolic logic and the advances in neuroscience, interest in the functional similarity between machines and the human brain has grown rapidly [8]. In 1943, McCulloch and Pitts [9] created a model of the biological nerve cell and revealed that any *Boolean* function could be implemented by a network of such artificial devices. They demonstrated that the brain is composed of a large network of neural units acting as pure and reliable ***logic gates*** [10].

In his book *Organization of Behavior*, published in 1949, Donald Hebb proposed the structural changes in the connections of nerve cells after learning [11]. Inspired by Hebb's rule, in 1958, Frank Rosenblatt invented the machine called *Mark I Perceptron*, the first Artificial Neural Network (ANN), which uses mechanically created artificial neurons to identify simple geometric forms [12]. Although being designed as ***binary*** devices simulating the behavior of nerve cells in the brain, ANNs evolved into ***analog*** models, and calculating real-valued ***weights*** of network connections became central to network training [13,14].

Along the historical progress, Artificial Neural Networks have been drifted away from being a mechanism that imitates the functioning of the human brain and turned into an advanced computational model to solve advanced classification problems. Recent ANN models lack many physiological features of the nervous system and possess fundamental limitations [15]. In addition to being biologically implausible, the major drawback of today's ANN models is the vast necessity of computation power to process extensive data. Most recently, highly complex architectures of deep learning approaches make it almost impossible for a typical researcher to train and test datasets such as ImageNet [16].

Contrary to Rosenblatt's analog approach, John Von Neumann proposed a digital computation architecture to simulate the learning process in the brain and process information with artificial nerve nets, popularly known as digital logic circuits of today's microprocessors [17–19]. Von Neumann emphasized the all-or-none character of biological neurons and argued that a neuron transmitting an impulse is comparable to a *binary digit*. This feature led to a general observation of the nervous system and its functioning as, in his words: ***prima facie digital***. He viewed each neuron as a *black box* and said that this organ receives digital stimuli from its inputs and emits digital stimuli as output. Von Neumann concluded that the stimulation of nerve pulses by appropriate combinations of other nerve pulses makes the neuron comparable to a ***logic gate***.

In this paper, we attempt to create a new kind of artificial neural architecture which will behave more like the human brain in a digital way and achieve similar advanced capabilities as the brain does. The new AI paradigm we propose simulates the process of neuroplasticity by creating an artificial structure that will generate augmenting neural networks by forming arbitrary connections and utilizing neurons in an evolutionary fashion, constantly optimized by Genetic Algorithms. The neurons in our model behave like logic gates and imitate their natural counterparts with an "*all-or-none*" character. Unlike Artificial Neural Networks, NLC is a weightless structure with no training procedure such as backpropagation. It is all about discovering neural architectures and evaluating their performance to find a learnable framework with generalization ability. The algorithm starts with minimal network architecture and complexifies it through evolution to simulate neuroplasticity. We evaluated our model on well-known binary classification datasets using advanced performance metrics and compared the results with modern and competitive machine learning algorithms.

The remainder of the paper is outlined as follows: In Section 2, we examine the theoretical background and review previous works from a computational neuroscience perspective. Besides, we explain the motivation behind our proposed AI paradigm and report several factors which inspired us in the way of this approach. In Section 3 we detail the methodology of NLC and explain how it is implemented. In Section 4, we report the experimental setup and analyze the results in Section 5. We conclude our paper in Section 6 together with intentions and recommendations for future work.

2. Theoretical background

At the beginning of the 20th century, Santiago Ramón y Cajal, the Spanish Neuroanatomist, also known as the *father of neuroscience*, proposed the ***Neuron Doctrine*** that learning is established in the brain with the modification of interneural connections (junctions) [20–22]. Later, Sherrington [23] named these connections as ***synapses*** [1]. In 1893, Eugenio Tanzi, the Italian Neuropsychiatrist, proposed that repetitive activity in a neuronal pathway through learning or practice can lead to the reinforcement of existing connections [24,25]. Finally, Jerzy Konorski [26], a Polish neurophysiologist was known to be the first person who coined the term ***neural plasticity*** and developed theoretical ideas about the learning process related to neuronal activity.

In the 1930s, the Russian physicist Nicolas Rashevsky proposed one of the first mathematical models of a nerve cell and showed that creating an artificial neuron is possible [4,27]. His theory was based on the *all-or-none* law of nervous activity, in which neurons either produce an impulse or remain at rest based on whether the summation of incoming pulses is above a certain threshold. He accepted the popular idea that there exist both excitatory and inhibitory connections among nerve fibers and finalized his theory based on the assumption that these quantities (e for excitatory and j for inhibitory) determine the frequency of impulses of efferent fibers [27]. His idea was to use a pair of linear differential equations and a nonlinear threshold operator to model nervous impulses [4–6,28,29]. Cull [29] summarizes his findings as:

$$\begin{aligned} &Input: = I(t) \\ &\frac{de}{dt} = AI(t) - ae \\ &\frac{dj}{dt} = BI(t) - bj \\ &Output: = H(e - j - \theta) \end{aligned} \tag{1}$$

where θ is the threshold and $H(x)$ as the Heaviside Step Function. Here, the excitation and inhibition of a neuron can be represented as e and j. The output of the Heaviside operator, either 0 or 1, will be able to model the *all-or-none* character of a biological nerve cell.

In 1943, McCulloch and Pitts formulated a theory of how networks of neurons express ideas in the form of *logical propositions* [9,27,30]. In their landmark paper "***A Logical Calculus of Ideas Immanent in Nervous Activity***", they employed Boolean logic and the mathematical notion of computation for the first time to explain how neural mechanisms might realize mental functions. They were the first to apply Boolean algebra to the behavior of nervous networks and the premise of their theory explained that excitatory or inhibitory signals are equivalent to logical propositions [27]. Knowing that each of the brain's nerve cells only fires after a kind of threshold has been reached, they become to realize that a neuron's all-or-none binary setup could make it work as a ***logic gate***, taking in multiple inputs and producing a single output. Variation of a neuron's threshold will obviously make it perform "AND", "OR" and "NOT" functions [31]. They called the first artificial neurons as Threshold Logic Unit (TLU), or Linear Threshold Unit, and used *Heaviside Step Function* as a transfer

function. Their simplified mathematical model was subsequently referred to by later researchers as a *McCulloch-Pitts (MP) formal neuron*. The MP neuron was a binary-state unit with its activation levels restricted to only *zero* and *one* [32].

An MP Neuron can be expressed with the following mathematical model: Let $u_i(t)$ denote the state of the ith neuron at a time t. Suppose $u_i = 1$ if the neuron is active, 0 otherwise. Let $H(x)$ be the Heaviside step function:

$$H(x) : = \begin{cases} 1 \rightarrow x > 0 \\ 0 \rightarrow x < 0 \end{cases} \tag{2}$$

Let time be measured in quantal units Δt, so that $u(t + \Delta t) = u(n\Delta t + \Delta t) = u(n + 1)$. Then the activation of a McCulloch-Pitts neuron can be expressed by:

$$u_i(n + 1) = H\left[\sum_j w_{ij} u_j(n) - v_{TH}\right] \tag{3}$$

where w_{ij} is the (assumed) weight of the $(j \rightarrow i)$th connection, and where v_{TH} is the activation threshold [30]. Obviously, activation will occur if and only if the total excitation $x = \sum_j w_{ij} u_j(n) - v_{TH}$ reaches or exceeds zero.

Cowan [30] defines McCulloch-Pitts (MP) neurons as synchronous binary elements or switches that can be combined in various circuits to implement logic gates (AND, OR, NOT, etc.). For example, activation of n binary excitatory inputs, no inhibitory inputs, and the internal threshold θ set to a positive value of less than one, the MP neuron will implement an OR gate. On the other hand, a neuron with n excitatory inputs stimulated, with no inhibitory inputs and threshold θ set to zero will yield an AND gate [32].

Contrary to the general perception of the digital character of the nervous system, Rosenblatt started the idea of "***analog weights***" between connections of neural units and formulated ANN output as the weighted sum of input signals transformed by an *activation function*. The training is carried out to find the best numerical values for the weights between neurons. An artificial neural network, having at least a hidden layer can theoretically approximate any non-linear continuous function. Because of this feature, it is considered as a *Universal Function Approximator* [33–37].

ANNs, where activation starts from the inputs and flows through hidden layers to the output, is defined as a *feedforward neural network*. Likewise, the output of a neuron in a feed-forward ANN can be calculated as the weighted sum of its inputs squashed with a transfer function:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{4}$$

where $x_1, x_2, \ldots, x_n$ are inputs, $w_1, w_2, \ldots, w_n$ are synaptic weights, b is the bias, and f is the activation function (e.g., sigmoid, tanh, etc.).

Biological Implausibility of Artificial Neural Networks

Unlike today's artificial neurons, biological neurons fire in discrete pulses. Although the original idea of McCulloch and Pitts was centered on a calculus of ***logic***, the question of why we insist on building artificial models on a decimal scale and forcing computers to make calculations to find derivatives of complex activation functions remain unanswered. Furthermore, connecting every neuron to another in fully connected ANNs stands as an improper wiring mechanism when compared to the sparse structure of biological networks. The findings of neuroscientists clearly show that biological neurons are not laid out in a fully-connected fashion and the structure of the nervous system is evidently not organized in layers, or at least, not in the way we use in ANNs [38].

It is so ironic that, after significant progress on ANNs, the direction of research went into finding similarities in biological facts and the computational paradigm. This backward move was motivated by the aim to prove (forcefully) the biological plausibility of the models developed, however, most of the attempts turned out to be unsuccessful. First, there is no convincing evidence that the information flow between biological neurons is running in an analog way. Neuron impulses have an all-or-none character and the action potentials generated are clear ***spikes*** of electrochemical transmissions. Some researchers tried to explain the ***weights*** as the repetition frequency of these action potentials, but they were unable to find how they relate to the behavior of neurons [39–41].

Secondly, how the brain implements the error back-propagation method, as employed in ANNs, is a key question that remains unanswered [42]. Research in the field of neuroscience suggests that *backpropagation* has no biological plausibility since there is no backward flow in our nervous system [43–45]. In the cerebral cortex, when a neuron fires, electrochemical signals flow through the axon and activate other connected neurons. Postsynaptic neurons do not fire this activation back to the presynaptic neuron [15,20].

The efficiency of Pruning, Skip Connections, and other Sparsification Methods

The major motivation for our efforts in this paper is the remarkable efficiency of *sparsification* methods developed in the last three decades. LeCun et al. [46] proposed one of the earliest destructive architecture optimization procedures, namely *Optimal Brain Damage* which removes unnecessary weights and minimizes the complexity of the network. Many groundbreaking achievements were made in the following years to explore optimum network architectures by using metaheuristics, reinforcement learning, and other approaches. Hinton et al. [47] introduced ***dropout*** which randomly omits several hidden units in a feedforward ANN, and observed a significant increase in the performance of the network. The authors also achieved success in reducing overfitting. The stochastic nature of dropout makes it somewhat similar to neuroplasticity in biological neurons. Once applied, dropout has the effect of modifying ANN layers with a different number of nodes on a diverse connectivity scheme. Furthermore, it breaks up situations where ANN layers co-adapt to correct errors from prior layers, thus making the model more robust.

Inspired by pyramidal cells in the cerebral cortex, He et al. [48] developed the *Deep Residual Learning Framework* (***ResNet***) and re-introduced the idea of skip connections for state-of-the-art deep CNN models. As the name suggests, *skip connections* skip a layer (or some layers) in the networks and feed the input of further layers or direct output. In order to ease the difficulty of training deeper networks, Srivastava et al. [49] proposed ***Highway Networks***, which was developed with the inspiration from *Long Short-Term Memory (LSTM)* recurrent neural networks using an adaptive gating mechanism to allow computation paths along which information can flow across many subsequent layers without loss. Similar to ResNet, this model also bypasses signals between layers using identity connections. As flexible and adaptive network architectures become popular, Cortes et al. [50] proposed ***AdaNet*** which removes the burden of network design from the user and learns architecture and weights automatically for a single task. Another way to implement skip connections is by concatenation of previous feature maps, which was proposed as ***DenseNets***, in another landmark paper by Huang et al. [51]. The authors introduced *direct connections* between any two layers

with the same feature-map size and obtained more compact models with feature concatenation.

Dendritic Neural Models

The conventional McCulloch-Pitts neuron model has long been criticized by researchers due to oversimplification of its calculation scheme using only weights [52]. Studies in the field of neuroscience identify biological aspects of real biological neurons with their temporal and spatial characteristics, and with the recent introduction of the spiking neuron model, a great potential has been discovered for solving more complicated deep learning tasks [53–55]. Furthermore, dendritic structure of biological neurons paved the way for more elaborate studies which exploit nonlinear nature to replace multi-layered artificial neural networks [56–58].

Lately, dendritic neural models (DNM) showed significant performance over challenging classification, as well as regression tasks and applied to several real-world problems as a competitive alternative to state-of-the-art deep learning models [59–61]. Although showing some similarities, our NLC model differs from DNM from a calculation perspective, since DNM mostly relies on backpropagation to optimize the final logic circuit. As explained in the next section, our proposed NLC model directly achieves biologically-plausible dendritic structures without an extensive calculation process. However, the success of recent DNM approaches clearly supports the baseline idea behind Neural Logic Circuits with great capabilities in terms of nonlinearity.

3. Neural logic circuits

In this work, we propose Neural Logic Circuits (NLC), as a novel evolutionary, weightless, and learnable neural architecture loosely inspired by the neuroplasticity of the brain. NLC can be defined as a neural network formed with a three-dimensional array of logic gates to simulate the neural circuitry of our nervous system. Unlike Artificial Neural Networks, NLC is a weightless structure with no training procedure such as backpropagation. NLC is all about discovering neural architectures and evaluating their performance to find a learnable framework with generalization ability. For this purpose, we employed an automated evolutionary architecture design method using Genetic Algorithms, which is a population-based global search algorithm inspired by *evolution theory* and the *survival of the fittest* principle. NLC uses training data to rewire its circuitry and modify its neural composition. All candidate solutions represent a network in population and their fitness is calculated by the classification accuracy on training data. Finally, the generalization ability of an NLC is evaluated by introducing test instances the network has not seen before.

In a nutshell, NLC starts with minimal network architecture and complexifies it through evolution to simulate the neuroplasticity of the brain. This is similar to natural evolution where species add new genes gradually to establish adaptation. Logic gates take multiple inputs and produce a binary output with an *all-or-none* character.

Representation

The problem of how a network architecture is genotypically represented plays a key role in the design of an evolutionary mechanism [62]. The early works mainly concentrated on innovative methods for developing the best chromosome representation procedure to express the ANN design in an encoded string [63–66]. Researchers mostly referred to biological facts from nature and proposed developmental patterns to minimize artificial genetic code. Fractals and L-Systems were mainly used as a source of inspiration to build connectivity matrices of ANNs (Weiß 1993,1997).

Table 1
Integer assignments for gate types.

Gate type	Assigned integer
AND	0
OR	1
NAND	2
NOR	3
XOR	4
XNOR	5

Here we adopt a direct encoding approach similar to the pioneering work of Miller et al. [68]. In this method, each network design is represented by a connectivity matrix mapped directly into a genotype, built with an array of bit strings. We propose variable-length chromosome encoding, which determines the size of the matrix depending on the number of gates used. Each entry on the connectivity matrix indicates the existence of neural connectivity between two units (*true (1)* if a connection exists and *false* (*0)* for no connection). Then, successive columns are concatenated to build the genotype.

Unlike conventional evolutionary approaches, we used two separate chromosomes (Chromosome A and Chromosome B) in synchronization with each other for every individual in the population. The first chromosome is used for gate types (neurons). The latter represents gate connections (synapses). In NLC, A logic gate simulates the behavior of biological neurons and can have six different types: AND, OR, NAND, NOR, XOR, and XNOR. In our implementation, each gate is represented with an integer number between 0 and 5.

Let $g_0, g_1, g_2, \ldots, g_n$ be neural units (logic gates) with integer variables, $g_i \in \{0, 5\}$. Table 1. shows integer assignments for each gate type.

The second chromosome (B), which represents the connectivity of neurons, contains only binary values, either *true* (1) or *false* (0). The evolutionary mechanism starts with a pre-determined number of gates, and the network is gradually augmented, starting from a minimal structure. A column is generated for each gate added to the network. Each row in this column has an entry for previous neural units, as well as sensory inputs. The first gate g_0 in the network has only rows for input units. The second gate g_1 has a row for g_0(to indicate connectivity), in addition to inputs. The last gate in the network is the *output* gate. Every gate, as well as the output gate, can have connections from all previous neural units, including the inputs. So, a direct connection from input to any unit is possible. A sample network with two inputs, five gates, and single output is explained in Fig. 1.

Unlike Artificial Neural Networks, an NLC network does not have layers. Neural units are distributed spatially in three dimensions, and there is no constraint in connectivity. Gates are placed in hierarchical order and represented in the chromosome in the order they are added to the network. The combinational circuit of the sample NLC network given above is illustrated in Fig. 2.

From a biological perspective, the brain is central to integrating sensory inputs and motor commands by utilizing its neural circuitry. The specific area of the brain called *the neocortex* is generally associated with intelligence and takes about 70% of the brain mass. The neocortex accommodates billions of nerve cells in several regions for various cognitive functions, such as visual, language, and touch. The sensory inputs in each region determine its purpose [38]. From a simplistic view, the NLC network constitutes tiny fractions of neocortex regions designed to perform particular tasks by taking inputs from the sensors and generating motor commands to muscles. A hypothetically generated image of a biological neural network corresponding to the sample NLC network given in Fig. 2 is illustrated in Fig. 3.

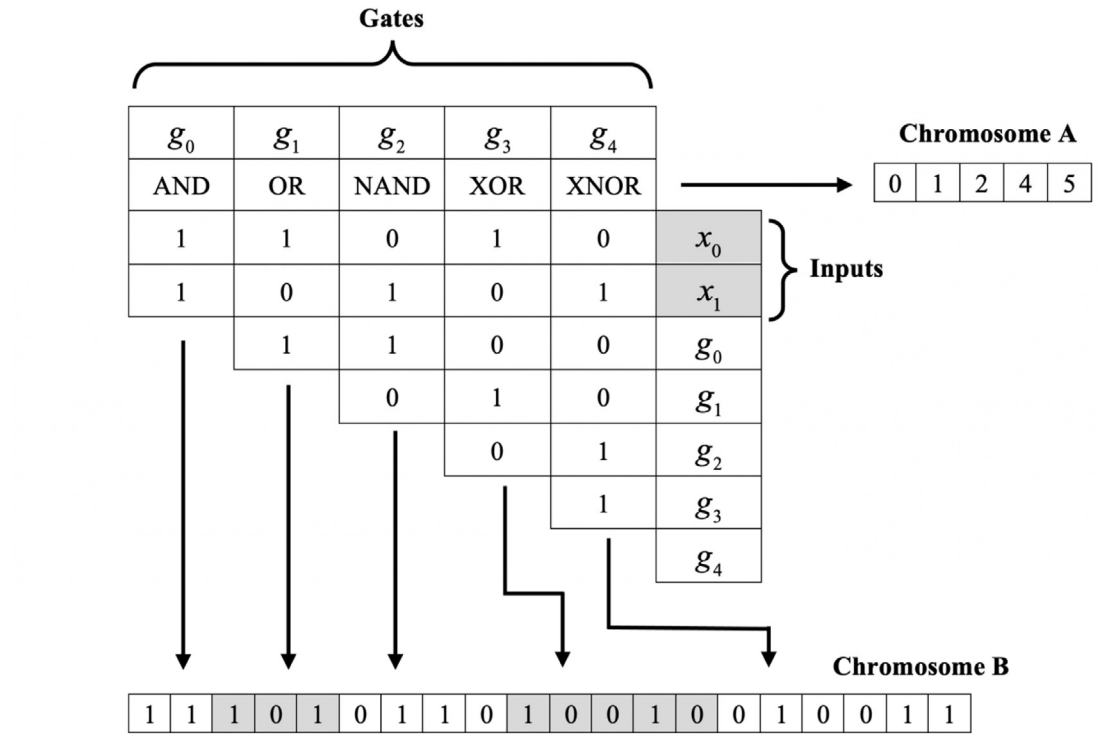


Fig. 1. A sample NLC network with two inputs, five gates, and a single-output (gate types are assigned randomly). Each column in the connectivity matrix represents a neural unit (logic gate). Binary values in the matrix show the existence of a connection between two units (row to column). Chromosome A is the array of integers assigned to gates in the network. Chromosome B is built with the concatenation of binary values for gate columns.

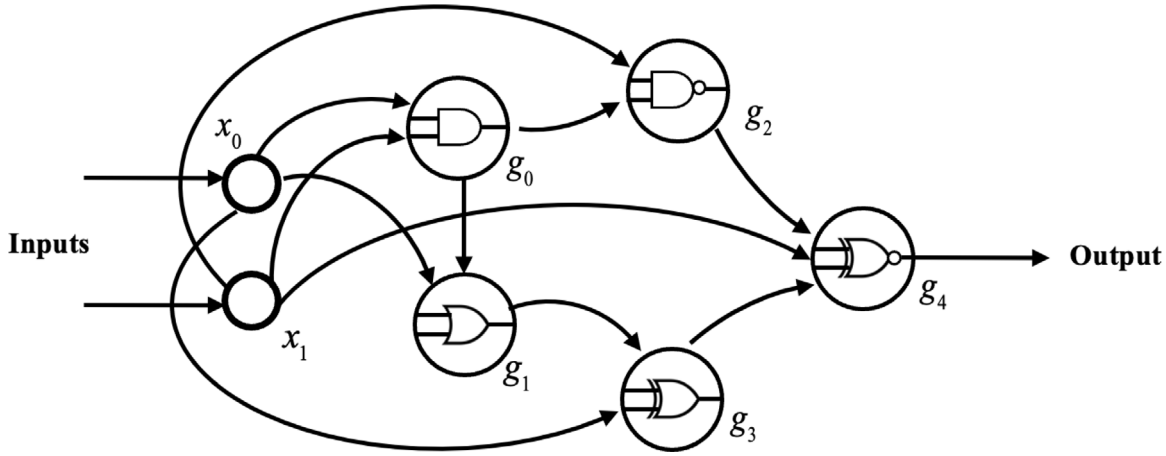


Fig. 2. The combinational circuit of the sample NLC network which was given in Fig. 1. Unlike Artificial Neural Networks, an NLC network does not have layers. Neural units are distributed spatially in three dimensions and there is no constraint in connectivity. Therefore, a direct connection from input x_1 to output is possible.

Calculation Flow

The output of an NLC network is calculated in a forward flow, starting from the sensory inputs and taking the outputs of each gate hierarchically. ***Unlike Artificial Neural Networks, an NLC network does not have layers, and has no backward flow.*** Similar to biological neurons, NLC neural units are distributed spatially in three dimensions with no connectivity constraint. As shown in Fig. 4., the output y_i of each gate g_i becomes available for the following gates if a connection exists.

Training and Evolution — Simulation of Neuroplasticity

In the real world, the human brain receives outside stimuli via biological receptors (eyes, ears, etc.) and processes or stores this information via neural units in the brain. As discussed in previous sections, repetition of such stimuli or reinforcement of learning activity helps rewire the synaptic connections, and create or eliminate neurons, thus enabling producing correct output for specific inputs. For example, reading and repeating the course topics will help you get higher grades if you are preparing for an exam the next day because neuroplasticity will modify connections between the nerve cells and prepare brain to give the correct output for possible questions.

Neural Logic Circuits simulate neuroplasticity and re-design its network with the help of sensory inputs. Like other machine learning paradigms, NLC utilizes training data to learn and gets evaluated with test data that the network has not been fed before. This proves its generalization ability and adaptation to changes in the environment. From a digital perspective, training of NLC can simply be described as designing the optimum structure of a

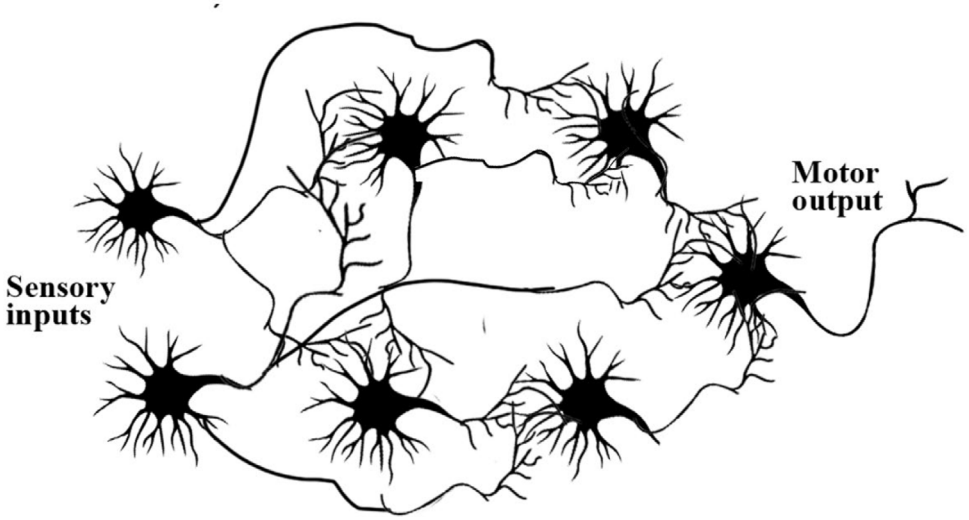


Fig. 3. A hypothetically generated image of the biological neural network corresponding to the sample NLC network is given in Fig. 2. Inputs are received from receptors (such as eyes, ears, etc.) and processed by nerve cells, thus producing a motor output (like muscles).

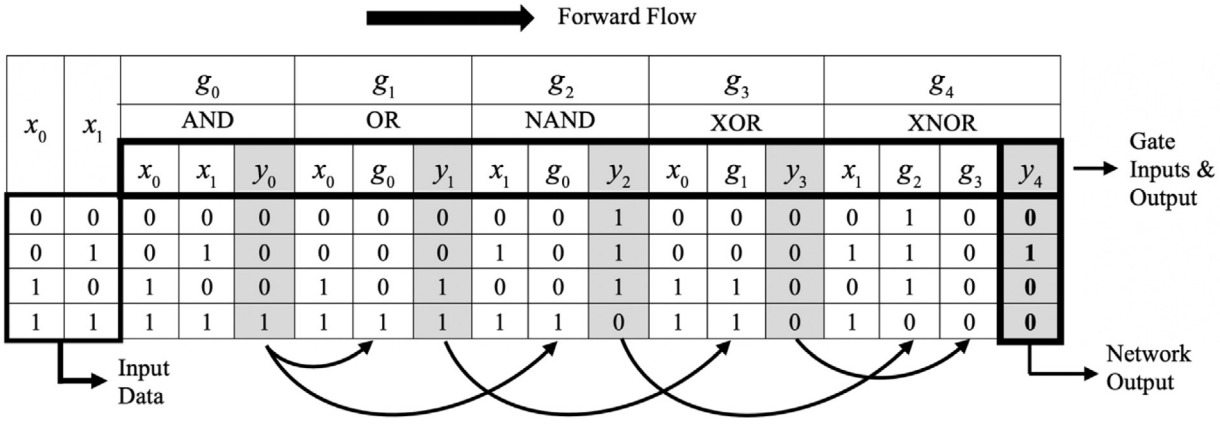


Fig. 4. Forward flow of the sample NLC network for all possible input values of x_i. Columns below each gate show the inputs and output for that gate. The last column y_i shows the calculated gate output. The final output y_4 from the gate g_4 gives the network output. Similar to Combinational Logic Circuits (CLC), the forward flow starts with the inputs and progresses hierarchically through gates in ascending order of indexes.

Combinational Logic Circuit (CLC). There are several approaches, as we know today, for automating the design of CLCs, including Karnaugh maps [69], Quine-McCluskey method [70,71], Espresso [72], and Multiple Level Logic Optimization System (MIS) [73]. Lately, heuristic approaches such as Genetic Algorithms have been utilized to create novel designs [74–78].

In this initial version of NLC, we utilized Genetic Algorithms (GA) as a competitive optimization tool to simulate neuroplasticity by automatically discovering neural structure and best connectivity. Genetic Algorithms are a class of population-based global search algorithms inspired by biological evolution with the *survival of the fittest* principle. It was introduced by Holland [79] and further developed by his student Goldberg [80]. It addresses optimization problems by imitating natural evolution through a series of operators such as selection, crossover, and mutation [81].

It is important to note that one can utilize any other optimization algorithm to find the best architectures for NLC. We determine to focus mainly on the baseline idea behind NLC rather than how it is optimized. Researchers are further encouraged to experiment with alternative approaches to best optimize NLC architectures.

Creating Initial Population

The first phase of a genetic algorithm is the generation of the initial population. Each individual in the population represents a solution to the problem. The *population size* is a crucial parameter that needs to be determined carefully before starting the algorithm. If the population is too small, finding near-optimal solutions will be faster but a broad search may not be possible. If the population is large, convergence will be slower, but better architectures can be obtained due to the diversity of search space. The generation of the initial population is usually carried out randomly. This allows starting from different points in the search space. One of the most significant advantages of Genetic Algorithms is their superior ability of parallelization. Genetic operations are carried out separately for each individual, bringing the flexibility for intensive computations.

Here, each individual in the population represents a complete NLC network. As an initial step, we generated the initial population for both chromosomes simultaneously. Then, we assigned random gate types for each neural unit and random binary strings for each column in the connectivity matrix. We introduced two essential parameters for comparing solution performance in the experiments. The first parameter is the *minimum gate count* ($g_{\min}$). Because the algorithm starts with a minimal structure,

Algorithm 1: Tournament Selection in NLC (v1)

Input: Population P_t , tournament size $t \in \{1,2,...,N\}$

Output: Q_t

$Q_t \leftarrow 0$
foreach *individual in* P_t **do**
  Randomly select N individuals from P_t and create subset $J_1,...,J_n$
  for $i \leftarrow 1$ **to** N **do**
    $J_i \leftarrow$ find individual with the highest fitness
  return J_i
  $P_{t+1} \leftarrow J_i$
  $Q_t \leftarrow P_{t+1}$
end
return Q_t

this value is set to 1 for many problems since we usually prefer to start from the most primitive network possible. For example, solving the non-linear XOR problem requires only one gate (an XOR gate), and the algorithm is most likely to converge on the first or second iteration of GA. However, more complex problems require complicated networks, and the user will be able to determine the minimum gate count manually. The second parameter in this step is the *maximum connections* ($c_{\max}$) between neural units. Biological facts suggest that there can be up to a thousand connections as an input to the dendrites of a nerve cell in the brain. However, sparse networks proved to be more efficient, and we wanted to give the user the flexibility to adjust the sparsity of the NLC. Although keeping this value set to the maximum gate count available, the user can limit the maximum number of connections and establish a sparser network to compare efficiency.

Fitness Evaluation

The *fitness function* evaluates the convergence of the individual solutions in a population. It provides a quantitative measurement that determines the probability of selection in the next iteration. A typical machine learning algorithm can simply be evaluated by its generalization ability by calculating classification accuracy on test instances. For this purpose, typically, the dataset is divided into two parts based on a train-test-split ratio. This value is usually set to 75% training data and 25% test instances for most datasets. Depending on the size of the data, test instances can be reduced to 5%. Another method for evaluating the performance of a network is the *k-Fold Cross Validation* method, which splits data into *k* number of subsets and provides a more robust estimate of the model's accuracy by measuring generalization on all *k* subsets of data. Then, the average performance of all subsets is calculated to obtain the general fitness of the individual solution.

In NLC, we used ***Stratified k-Fold Cross-Validation***, which stratifies each subset, meaning that all folds have the same ratio of *true* and *false* outputs as the dataset. Each candidate network in an individual solution is evaluated using this method, and the average of all k-folds is used as the fitness value.

Selection

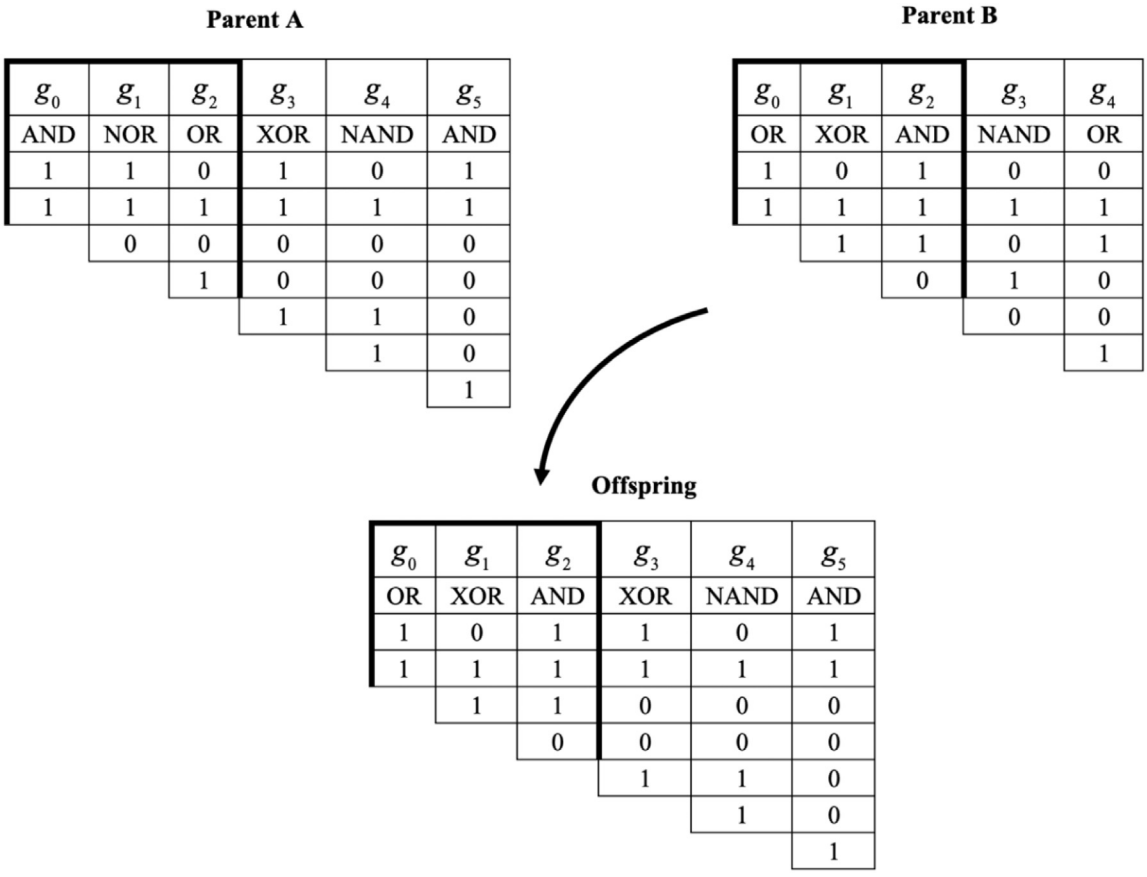
The selection phase eliminates low-quality individuals and transfers better networks to the next step, reproduction. Based on the Darwinian principle of *survival of the fittest*, individuals with higher fitness are more likely to win during selection, although the process involves randomness in nature. There are several methods developed for selection in GAs. Among them, the roulette wheel, tournament selection, and rank methods are the most commonly used.

Our evolutionary search procedure is based on *tournament selection,* executing this phase in two steps. First, it randomly selects a pre-determined amount of individuals from the entire population (tournament size). Then, among this subset, it returns the individual with the highest fitness as the tournament winner. The algorithm repeats this process with a number of tournaments as the population size (Algorithm 1).

Crossover

The reproduction phase is executed by a *crossover* operator, which simulates the sexual generation of a child (offspring) from two parents [82]. Individuals selected for crossover produce offspring(s) who share the common characteristics of their parents. In GA, it is accomplished by replacing variable subsets of the binary strings in the chromosome of two-parent individuals selected according to a crossover probability P_c. Extensive research proves the efficiency of this procedure, which alters the sampling rate of hyperplanes in the search space by recombination of parent genes. Crossover helps speed up the discovery of high-quality solutions to the problem by combining valuable pieces of genetic information called *building blocks* [83,84]. How the crossover is carried out may vary depending on the encoding structure and the problem's nature. The most commonly used crossover methods are single-point, two-point, arithmetic, and uniform crossover.

In conventional neuroevolutionary approaches, the crossover is mostly avoided due to loss of functionality by producing invalid networks in the offspring. However, there are also works in the literature which successfully applied crossover to build ANN architectures using evolutionary approaches. In NLC, crossover plays a key role in discovering optimal solutions in unvisited regions of the search space. We applied crossover on both chromosomes in sync with each other and adopted the *single-point crossover* technique, which is known as a common and faster method. We started from the first individual, then iterated through the population by checking a random number each time to ensure crossover probability grants implementation. Then,



g_0	g_1	g_2	g_3	g_4	g_5
AND	NOR	OR	XOR	NAND	AND
1	1	0	1	0	1
1	1	1	1	1	1
	0	0	0	0	0
		1	0	0	0
			1	1	0
				1	0
					1

g_0	g_1	g_2	g_3	g_4
OR	XOR	AND	NAND	OR
1	0	1	0	0
1	1	1	1	1
	1	1	0	1
		0	1	0
			0	0
				1

g_0	g_1	g_2	g_3	g_4	g_5
OR	XOR	AND	XOR	NAND	AND
1	0	1	1	0	1
1	1	1	1	1	1
	1	1	0	0	0
		0	0	0	0
			1	1	0
				1	0
					1

Fig. 5. Single-point crossover operation on NLC (v1). The variable-length chromosome is divided into two equal parts on the connectivity matrix, and the first half of the gates are taken from other parents. This procedure avoids invalid networks being generated as offspring.

we took current and consecutive population member as parents (p_i & p_{i+1}). Since we utilize a variable-length encoding, we checked the lengths of both parents (say, parent A and parent B) and determined crossover points based on gate counts. The aim is to divide the chromosome of parent A into two equal parts in the middle, then override the first half with the gates of parent B having the same index. For example, if parent A has six gates and parent B has five gates, then parent A takes the first three gates (6 divided into 2) from parent B (Fig. 5) (Algorithm 2).

In Fig. 5, this is clearly indicated. Above, it shows crossover operation between a 6-gate network (parent A) and a 5-gate network (parent B). We take the first three gates from parent B, place them in the first part of the offspring, then take g_3-g_4-g_5 from parent A and place them as the g_3-g_4-g_5 of the offspring. The outcome is the offspring network, and the chromosome is formed by the concatenation of the resulting connectivity columns. This avoids any loopback by crossover operation and always produces valid networks.

However, there might not be enough gates on parent B to fill the first half of parent A. In that case, the algorithm defines the crossover point as the gate count of parent B. For example, if parent A has six gates and parent B has only two gates, then the crossover point will be at gate 2 since there are only two gates at parent B, which is less than half the number of gates at parent A.

Algorithm 2: Single-point Crossover for NLC (v1)

Input: Population P_t, probability of crossover P_c
Output: Population after crossover; Q_t

$Q_t \leftarrow 0$
foreach *individual* p_i **in** P **do**
  $r \leftarrow$ Randomly generate a number from (0,1)
  if $r < p_c$ **then**
    Determine number of gates to exchange
    $o_1 \leftarrow$ take the gates on the first half of p_{i+1}
    $o_2 \leftarrow$ take remaining gates from p_i
    $Q_t \leftarrow o_1 \cup o_2$
  else
    $Q_t \leftarrow p_i$
  end
end
return Q_t

Mutation

In order to enhance genetic diversity, the *mutation* is implemented in Genetic Algorithms by modifying random components of the chromosome in an amount determined by the mutation

rate P_m. Mutation in nature is a change or degradation of a DNA molecule found in the nucleus of the living cell and enables the emergence of hereditary properties. Some possible causes of mutation are radiation, X-ray, ultraviolet, sudden temperature changes, and degradation due to chemistry. In living organisms, the mutation is rare and occurs in very small regions of the chromosome.

In Genetic Algorithms, mutation helps improve solution quality by occasionally introducing minor structural changes in the population. It prevents the algorithm from converging to a local minimum and enables exploring a broad search space to find the optimal solution. In NLC, mutation simulates biological rewiring in the neural circuitry by slightly modifying connectivity or neurons. In this phase, NLC renovates several network gates, which are randomly selected in an amount determined based on the mutation rate. This renovation is applied both on the gate type and the connectivity string (Fig. 6) (Algorithm 3).

Algorithm 3: Implementing Mutation on NLC (v1)

Input: Population P_t, probability of mutation P_m
Output: Population after mutation; Q_t

$Q_t \leftarrow 0$
foreach *individual* p_i **in** P_t **do**
  $r \leftarrow$ Randomly generate a number from (0,1)
  if $r < p_m$ **then**
    Randomly choose N gates from p_i, $G \in \{1,2,...,N\}$
    foreach *individual* g_i **in** G **do**
      Set a random gate type l, $l \in (0,5)$
      Initialize random connectivity
      $g_{mutated} \leftarrow$ replace existing gate
    end
    $p_i \leftarrow G$
  $Q_t \leftarrow p_i$
  else
  $Q_t \leftarrow p_i$
  end
return Q_t

Elitism

Due to its stochastic nature, Genetic Algorithms may degrade some near-optimal solutions during interim generations by applying crossover and mutation operators. In order to prevent such situations, elitism is applied to the algorithm to preserve the best individuals and carrying them on to the next generation without change. This will protect the fittest solutions from being damaged by genetic operations and increase the algorithm's effectiveness. The number of elite individuals which will be passed to the next generation is controlled by a user-specified parameter called the *elitism rate* P_e. Research suggests that elitism improves the performance and convergence of GA in both single and multi-objective applications [85–87]. However, care should be taken to set this parameter as small as possible since it can cause early convergence and trapping to local minima. Setting the elitism rate higher will also affect genetic diversity adversely [88,89].

In NLC, we implemented elitism on a relatively small subset of the population and carried the best networks to the next generation to protect optimal networks from disappearing. Better solutions in successive generations can also replace those preserved solutions. This allows high-performing neural circuits to pass their traits to other candidate solutions, thus enabling the finding of near-optimal networks in a shorter time.

Augmentation

NLC starts from a minimal structure and gradually augments its neural circuitry by adding new neurons and introducing new connections. The biological plausibility of this phenomenon originates from *neuroplasticity* in the brain, which manifests itself, especially during early childhood. The idea draws its inspiration from extensive studies in neuroscience, revealing that the number of synapses in the brain is rapidly increased from birth to 3 years old. Augmenting topologies in Artificial Neural Networks is a widely accepted idea, successfully implemented in prominent works such as *NEAT* by Stanley and Miikkulainen [90] and, more recently *AdaNet* by Cortes et al. [50]. This also helps minimize the dimensionality of search space and reduce computational costs.

NLC sorts the population in each iteration based on their fitness and replaces the lowest-quality solutions with newly generated chromosomes, consisting of more gates. These new individuals are introduced to the population as random networks with no *a priori* information about the problem. Despite their potentially lower fitness as a disadvantage to other individuals, genetic operations such as crossover and mutation help combine their powerful features and increase the level of diversity, thus creating better networks in later generations. The number of population members, introduced as augmentation, and the speed of augmentation are also user-controller parameters called *augmentation rate* λ and *augmentation speed* ϕ. Fig. 7. shows the replacement of the current population with augmented individuals.

4. Experiments

In this section, we provide a detailed description of the experiments performed to evaluate the performance of NLC. To validate the efficiency of our proposed paradigm, we used several widely-used binary classification datasets from the UCI database and utilized advanced accuracy metrics, which further helped compare results with state-of-the-art machine learning methods. In order to simplify the presentation in this initial model (v1), we restrict our attention to the case of binary classification. The ultimate goal of a binary classification problem is to develop an intelligent model that makes estimations, where the result to predict, takes one of just two possible values (yes or no, true or false, etc.).

Benchmark Datasets

We conducted experiments on four public binary classification datasets taken from the University of California at Irvine (UCI) Machine Learning Repository: *Pima Indian Diabetes*, *Haberman's Survival*, *Cleveland Heart Disease,* and *Mammographic Mass*. These Biomedical datasets are generally associated with high-dimensional features with an imbalanced character. An *imbalanced classification* can be defined as a problem when class distribution is unequal. In such datasets, instances of data across known classes can be either biased or skewed. Hence, they are challenging models for almost all machine learning algorithms, resulting in poor estimation performance. To improve the reliability of our approach, we also conducted additional experiments on a relatively new and complicated *NBA Rookie Longevity Dataset* taken from data.world. Here is some detailed information about these datasets.

- *The Pima Indian Diabetes* dataset [91] was provided by the National Institute of Diabetes and Digestive and Kidney Diseases over female patients at least 21 years old of Pima Indian heritage. The aim of the dataset is to predict whether or not a patient has diabetes based on specific diagnostic factors. It is an unbalanced

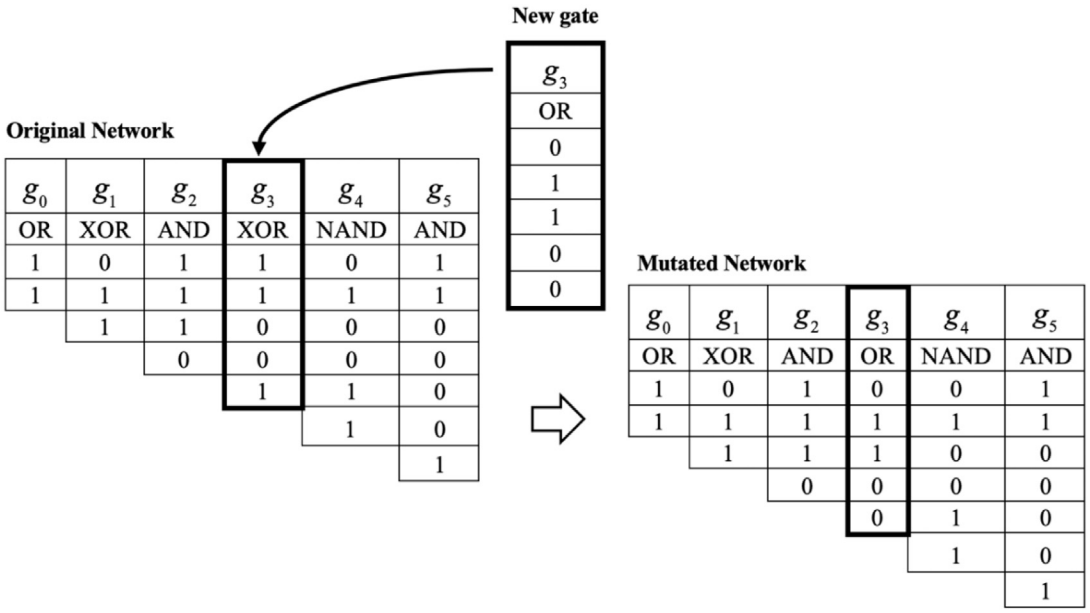


Fig. 6. Implementing mutation operator on NLC (v1). Randomly selected gates are replaced with a newly created gate. Mutation can change both the gate type and the connectivity string.

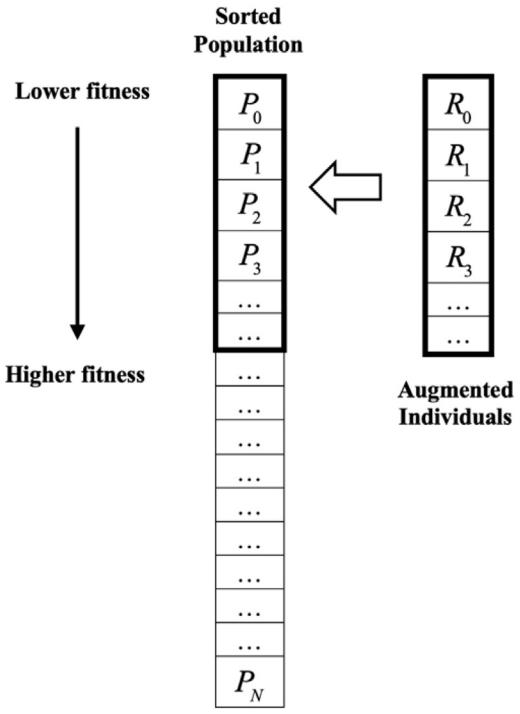


Fig. 7. Augmentation in NLC (v1). On each generation, the entire population is sorted based on their fitness, and a subset of lower-fitness individuals are replaced with newly created networks with gradually increasing number of neural units.

dataset that contains 768 instances with eight input variables. They are; the *number of times pregnant*, *plasma glucose concentration after 2 h in an oral glucose tolerance test*, *Diastolic blood pressure* (mm Hg), *Triceps skinfold thickness* (mm), *2-Hour serum insulin* (μU/ml), *Body Mass Index* (weight in kg/(height in m)^2), *Diabetes Pedigree Function* and *Age* (years). It has two classes as being tested positive (class 1) or tested negative (class 0) for diabetes. The positive class has 268 rows, and the negative class has 500 rows, respectively [92].

- The Haberman's Survival Dataset is another imbalanced binary classification dataset conducted by the University of Chicago's Billings Hospital, containing patient survival data who had undergone breast cancer surgery between 1958 and 1970. The data has 306 instances with three attributes. Input variables are the *patient's age at the time of operation*, *the year of operation*, and the *number of axillary nodes detected*. The class attribute is the patient's *survival status* within five years of surgery. Class 1 (survival) has 225 instances, and class 2 (non-survival) has 81 instances.

- The Cleveland Heart Disease Dataset is a subset of patient data collection from Cleveland, Hungary, Switzerland, and Long Beach [93]. Out of these four sets, Cleveland was mainly used as a benchmark by researchers in the field of machine learning. It contains 76 attributes (including the class attribute), but only a subset of 14 attributes are generally utilized to evaluate classifier performance. The class attribute has a binary value indicating the presence of heart disease. Data comprises 303 instances, with six samples having missing values.

- The Mammographic Mass Dataset is a collection of digital mammograms from female patients at the *Institute of Radiology of the University of Erlangen–Nuremberg* between 2003 and 2006. It was created to predict the severity of a mammographic mass lesion from BI-RADS attributes and the age of the patient [94]. The work aims to reduce the number of unnecessary biopsies for screening breast cancer. Benchmark data contains six attributes, including BI-RADS assessment (non-predictive), the patient's age, and three BI-RADS attributes (mass shape, mass margin, mass density). In a peer-review procedure, physicians provide a BI-RADS score for each incident, ranging from 1 (certainly benign) to 5 (strongly suggestive of cancer). Sensitivities and associated specificities can be estimated if all instances with BI-RADS assessments greater than or equal to a given value (ranging from 1 to 5) are malignant and the remaining cases are benign. These can be used to determine how well an ML algorithm performs compared to radiologists. The dataset has 961 records in total, with 516 benign and 445 malignant instances.

- *NBA Rookie Longevity Dataset.* NBA is the best basketball league in the world, and entry into this league is highly competitive. Only 1% of players from NCAA Division I colleges get selected by the NBA. Newly selected players must continue demonstrating their value on the court to stay in the league. ML models aim to determine whether a player will play in the league for five years by using a variety of rookie-year statistics. The dataset was provided by ***data.world*** and was last updated in 2017. It contains statistical information about 1340 players with 19 attributes such as games played, and average points per game. The class attribute has 831 positive (played in the league for five years) and 509 negative instances (62% positive, 38% negative).

Converting Datasets to Binary

As discussed in the previous sections, the binary system is the internal language of NLC. Prior to conducting experiments, we converted all data in the datasets to binary. For this purpose, we analyzed every input variable by grouping them into several categories. First of all, we used *one-hot-encoding* to convert categorical data to binary. Then, we converted integer and real-valued data into binary directly and determined the length of the input column from the maximum value of each variable. If the input contains negative values, we added a sign bit. It is essential to keep the length of each column fixed, and we aimed to minimize input dimensions while keeping the network sparse. Here is an illustration of how we performed this step with examples:

- Consider converting the 'Pregnancies' attribute to binary in Pima Indian Diabetes dataset. The maximum value is 17. When we convert 17 to binary, we obtain '10001'. The length is 5 bits. Then, we convert each input into 5 digits. For instance, if the attribute has a value of 3, the input column takes the value of '00011' (not '11').

- For floating numbers, first, we convert the value to plain decimals. For example, consider converting the 'BMI' attribute to binary in Pima Indian Diabetes. The maximum value for this attribute is 67.1. Here the precision is 0.,1 and we multiply the values by 10 to obtain plain decimal numbers. In this case, we multiply 67.1 by 10 and get 671. We convert 671 to binary and get '1010011111'. The outcome has 10 digits. If we encounter a negative value in the attribute, we add a sign bit, and the input column becomes 11 bits (it was not the case for BMI). Similar to the previous explanation, we convert each entry in the dataset to binary and make it 10 bits by adding zeros in front of it.

Despite similar approaches, binary conversion cannot be considered a pre-processing step in NLC. Allocating a fixed number of bits to an attribute will not cause a data leakage as in data pre-processing procedures like standardization. However, when working with actual data, researchers are encouraged to increase the length of the input columns and take universal maximums to handle greater values that may be encountered during the testing phase.

Peer Competitors

We examined recent works and selected several state-of-the-art machine-learning approaches that reported promising classification accuracies on the chosen benchmark datasets. These are Logistic Regression (LR), Random Forests (RF), ANNs, Decision Trees (DT), Naïve Bayes (NB), k-Nearest Neighbors (k-NN), and Support Vector Machines (SVM).

Experimental Setup

The proposed NLC paradigm is implemented using C++ programming language in native code[1] without any framework, while other competitor ML algorithms are experimented with using various ML frameworks (Keras, Tensorflow, etc.) on Python.[2] For better reproducibility of the results, we utilized constant seeds for randomized numbers. The source codes are also available on ***GitHub*** for reproducing experimental results reported in this paper.[3]

All initial parameter settings for genetic algorithms and peer competitors are chosen based on the conventions in the communities of evolutionary approaches and machine learning. Then, we further improved their efficiency with the hyperparameter optimization process and fine-tuned the best parameters of each algorithm for each dataset. Finally, in order to avoid data leakage, we applied data pre-processing to train sets only by using ***Pipeline*** class.

Table 2
Optimized hyperparameters for NLC.

Category	Hyperparameter	Symbol	Values
Network	Min gates	$g_{\min}$	1
	Max connections	$c_{\max}$	4
	Augmentation ratio	λ	5%
	Augmentation speed	ϕ	1.0005
	k-Folds	k	8–10
GA	Population size	N	100
	Tournament size ratio	t	5%
	Probability of crossover	P_c	0.55
	Probability of mutation	P_m	0.1
	Elitism ratio	P_e	0.03

Hyperparameter Optimization

Similar to all conventional machine learning approaches, NLC has several variables that affect the results and determine the algorithm's efficiency. Furthermore, genetic algorithm parameters contribute to the success of NLC in terms of accuracy and speed. Therefore, we applied a generic optimization process to find the best parameters. Hyperparameters and their values used in the experiments are given in Table 2.

For a fair comparison, we applied *Grid Search* and found the best hyperparameters for peer ML Algorithms. This resulted in the best results on each ML approach. Tables 3 to 9 show the fine-tuned hyperparameters on peer ML algorithms, respectively.

5. Results and discussion

The experimental results of the NLC and its peer competitors are given for each dataset in Tables 10 to 14, respectively. Tabulated data provides accuracy, sensitivity, specificity, precision, F1-Score, and balanced accuracy of each algorithm on separate columns, while mean and standard deviation values are reported on sub-columns under those metrics. For simplicity, we converted numerical results to percentages and rounded the values down to two digits. Researchers may refer to raw data stored at ***GitHub***.[4]

As expected, ML Algorithms show a moderate performance on Pima Indian Diabetes since it is an imbalanced classification dataset having 500 negative and 268 positive classes (65%

[1] Complete source codes for NLC are available at:
https://github.com/htanerunal/nlc/blob/main/nlc_pima/main.cpp
https://github.com/htanerunal/nlc/blob/main/nlc_haberman/nlc_haberman.cpp
https://github.com/htanerunal/nlc/blob/main/nlc_cleveland/nlc_cleveland.cpp
https://github.com/htanerunal/nlc/blob/main/nlc_mammograph/nlc_mammograph.cpp
https://github.com/htanerunal/nlc/blob/main/nlc_nba/nlc_nba.cpp.

[2] Complete source codes for peer competitors are available at:
https://github.com/htanerunal/nlc/tree/main/peer_competitors/pima
https://github.com/htanerunal/nlc/tree/main/peer_competitors/haberman
https://github.com/htanerunal/nlc/tree/main/peer_competitors/cleveland
https://github.com/htanerunal/nlc/tree/main/peer_competitors/mammograph
https://github.com/htanerunal/nlc/tree/main/peer_competitors/nba.

[3] Researchers may reproduce all results by using the source codes available at:
https://github.com/htanerunal/nlc.

[4] All raw results are available at:
https://github.com/htanerunal/nlc_results/raw/master/NLC_Results_Table_Sep_22.xlsx.

Table 3
Optimized hyperparameters for Logistic Regression (LR).

Dataset	*C*	*max_iter*	*Penalty*	*Solver*
Pima[a]	1	100	L2	Newton-cg
Haberman's[b]	0.1	100	L2	Liblinear
Cleveland[c]	0.0001	100	L2	Liblinear
Mammographic Mass[d]	0.1	100	L2	Newton-cg

[a]Source code available at:
https://github.com/htanerunal/nlc/tree/main/hyperparameter_search/pima.
[b]Source code available at:
https://github.com/htanerunal/nlc/tree/main/hyperparameter_search/haberman.
[c]Source code available at:
https://github.com/htanerunal/nlc/tree/main/hyperparameter_search/cleveland.
[d]Source code available at:
https://github.com/htanerunal/nlc/tree/main/hyperparameter_search/mammograph.

Table 4
Optimized hyperparameters for Random Forest classifier (RF).

Dataset	*n_estimators*	*Criterion*	*max_depth*	*max_features*
Pima	200	Gini	8	log2
Haberman's	100	Entropy	4	log2
Cleveland	300	Gini	6	log2
Mammographic Mass	300	Gini	4	Sqrt

Table 5
Optimized hyperparameters for Artificial Neural Networks (ANN).

Dataset	*Optimizer*	*batch_size*	*Epochs*
Pima	RMRprop	10	50
Haberman's	RMSprop	20	150
Cleveland	Adam	50	100
Mammographic Mass	Nadam	10	100

negative class). Therefore, a dummy classifier, which outputs false to every input, will at least get 65% accuracy. As a result, it is relatively easier to predict the negative class, and the specificity (true negative rate) of each algorithm turns out to be around 80%–90% accurate. On the contrary, they all perform poorly in the positive class (sensitivity), reaching the best at 60%. Our proposed paradigm, NLC, performed significantly better in all metrics, proving to be a competitive predictor on this task. General performance indicators such as F1-Score and balanced accuracy are also superior to other algorithms.

In another benchmark, Haberman's Survival Dataset, models are usually not expected to generalize since it is an extremely imbalanced dataset having 81 negative and 225 positive classes (73.5% on survived). As such, our experimental results confirm the poor performance of the peer competitors. Based on the results given in Table 11, conventional ML algorithms stand as ineligible to be used as a classifier since they cannot estimate the positive class at all (the best performs at 20.83% on sensitivity). Although plain accuracy figures seem moderate, each peer competitor's overall balanced accuracy is around 50%, which shows that they are no different from tossing a coin to predict the result. It is interesting to see, even for us, that NLC performed exceptionally well on this dataset. The F1-Score is 90.63%, and balanced accuracy is 75.68%. This clearly shows that NLC can learn from fewer data and can reliably be used in datasets with an imbalanced character.

Cleveland Heart Disease is also a small dataset with 303 samples, but it is relatively easier for ML models due to its target class with more balanced figures, with 165 on the positive and 138 on the negative classes (54.5% positive). As shown in Table 12, peer competitors perform relatively better on this dataset, with an average of 82% success on balanced accuracy. The best is the kNN algorithm, with 83.11% balanced accuracy. Again, NLC outperforms all peer competitors in every metric, achieving an impressive 85.75% on balanced accuracy and 87.71% on F1-Score.

Lastly, the experimental results for Mammographic Mass, which is a more extensive dataset compared to the previous two benchmarks, show good performance with peer competitors and prove the predictive capability of each model. This dataset contains 830 samples (without rows with missing values) and has a balanced distribution in the target class (402 positives, 427 negatives). Among peer competitors, the ANN gains the best performance in terms of balanced accuracy (80.96%). Our proposed NLC proves its superiority again, outperforming the best on this dataset on almost all metrics. NLC achieved 81.58% on balanced accuracy and 82.03% on F1-Score, showing that it is a highly competitive model on binary classification tasks.

The results for NBA Longevity Dataset are shown in Table 14. Peer ML models show a moderate performance in predicting the outcome of this challenging dataset. The best models among peers were Logistic Regression on accuracy (71.19%), Random Forest on Balanced Accuracy (67.66%), and SVC F1-Score (78.16%). Once again, NLC outperformed all peer competitors on all metrics with a remarkable superiority in accuracy and F1-score. The balanced accuracy of NLC is also slightly better than the best ML model evaluated.

To summarize, our experimental results reveal that on virtually all of these datasets, the proposed NLC paradigm outperforms all existing methods in terms of accuracy, F1-score, and balanced accuracy. Its remarkable performance on F1-Score and balanced accuracy clearly indicates that NLC not only discriminates successfully on positive classes but is also superior in predicting the negative classes. This is extremely important when selecting a model with imbalanced class distribution, especially in medical data where predicting minority class is crucial. The robustness of NLC also manifests itself in the smaller difference between accuracy and balanced accuracy. While other ML approaches are sometimes good at achieving high scores on plain accuracy, their balanced accuracy and F1-Scores are, most of the time, far from the nominal figures.

Further Discussion

Although our experimental setup on NLC is open to complex neural networks, the resulting neural structures are somewhat surprising since NLC often found very sparse networks with very few numbers of logic gates and connections. The Genetic Algorithm converged on the best results and remained at the same structures even after 3.000 iterations. Changing the initial parameters or tweaking GA variables did not change the overall result significantly. For example, starting to build the network with 100 gates converged on similar (sometimes worse) accuracy figures compared to starting with only one gate. Furthermore, changing the maximum number of connections between the neural units did not dramatically change the result either. The algorithm preferred to keep connections to a minimum, which helped to expedite convergence. Fig. 8 shows the neural network that NLC has built on the Pima Indian Diabetes task.

Experiments with NLC on Pima Indian Diabetes Dataset demonstrated that a network with only one gate (an AND gate) and two connections from the fourth and eighth binary values of the ***Insulin*** attribute gained superior accuracy on the test set of a cross-validation fold, achieving 81.63% F1-Score and 85.46% balanced accuracy. As previously reported, high performance was consistent on other CV folds, showing that the discovery of such neural connectivity gives a clue to our brain's remarkable capability of neuroplasticity.

Similar results were obtained from the experiments on other datasets. Fig. 9 shows the resulting network that NLC converged after 525 iterations on Haberman's Survival Dataset.

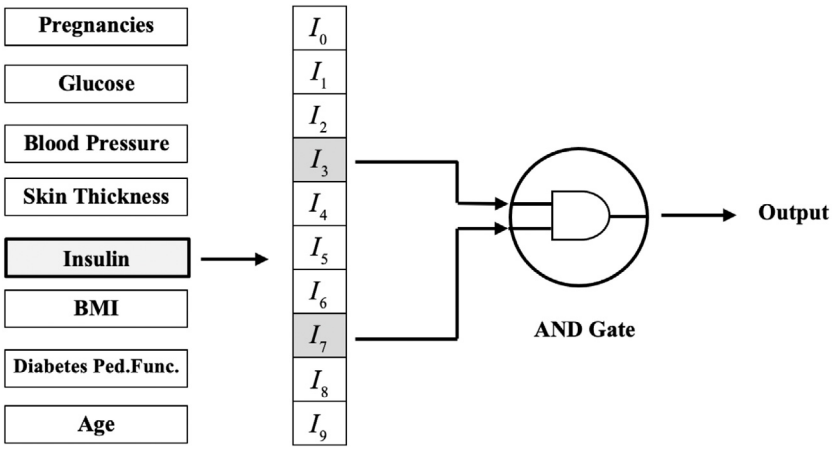


Fig. 8. The resulting NLC Network on Pima Indian Diabetes dataset. It has only one neural unit (an AND gate) and two connections. The one and only neural unit take its inputs from the fourth and eighth digits in the binary values of the Insulin attribute. The network achieved 88.15% accuracy on the test set of a cross-validation fold, with an F1-score of 81.63% and balanced accuracy of 85.46%.

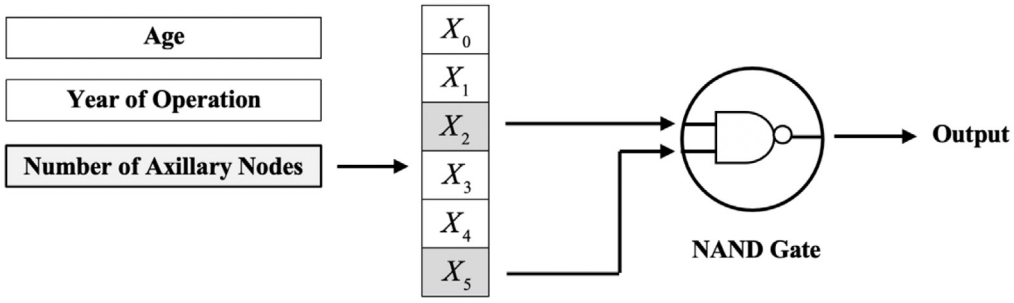


Fig. 9. The resulting NLC Network on Haberman's Survival dataset. The network has one neural unit (a NAND gate) and two connections. The only neural unit takes its inputs from the third and sixth digits in the binary values of the ***Number of Axillary Nodes*** attribute. The network achieved 86.65% accuracy on the test set of a cross-validation fold, with an F1-score of 90.48% and balanced accuracy of 86.93%.

Table 6
Optimized hyperparameters for Decision Tree classifier (CART).

Dataset	*Criterion*	*max_depth*	*max_features*	*min_samples_leaf*	*min_samples_split*
Pima	Gini	5	log2	9	3
Haberman's	Entropy	2	log2	3	3
Cleveland	Gini	5	Sqrt	9	6
Mammographic Mass	Gini	5	log2	7	4

Table 7
Optimized hyperparameters for Naïve Bayes classifier (NB).

Dataset	*var_smoothing*
Pima	0.432876128
Haberman's	0.005336699
Cleveland	0.533669923
Mammographic Mass	1.0

Table 8
Optimized hyperparameters for K-Nearest Neighbor classifier (k-NN).

Dataset	*leaf_size*	*Metric*	*n_neighbors*	*p*	*Weights*
Pima	1	minkowski	19	1	Distance
Haberman's	35	chebyshev	27	1	Uniform
Cleveland	1	minkowski	14	2	Uniform
Mammographic Mass	21	minkowski	9	1	Uniform

Table 9
Optimized hyperparameters for Support Vector Classifier (SVC).

Dataset	*C*	*gamma*	*kernel*
Pima	10	0.01	rbf
Haberman's	1	0.1	rbf
Cleveland	100	0.001	rbf
Mammographic Mass	10	0.1	rbf

Most surprisingly, on a dataset in which peer competitors are not even able to generalize, NLC found a superior network with only one gate (a NAND gate) and two connections. Furthermore, the last attribute of the dataset, the ***Number of Axillary Nodes*** contributed as the inputs to that gate, and the resulting sparse network achieved 86.65% accuracy on the test set of a cross-validation fold, with an F1-score of 90.48% and balanced accuracy of 86.93%.

On the Cleveland Heart Disease dataset, a similar network with only one gate outperformed all peer competitors in all evaluation metrics. The resulting NLC network is shown in Fig. 10.

Once again, NLC built a very sparse network using two attributes (***ca*** and ***thal***) as inputs and achieved 91.89% accuracy on the test set of a cross-validation fold, with an F1-score of 92.68% and balanced accuracy of 91.62%.

In the Mammographic Mass dataset, NLC built a similar network with one gate (OR gate) and two connections from two attributes (***shape*** and ***margin***) of the dataset (see Fig. 11).

Table 10
The classification results of the proposed NLC paradigm against the peer competitors on the Pima Indian Diabetes dataset.

	Accuracy (%)		Sensitivity (%)		Specificity (%)		Precision (%)		F1-Score (%)		Balanced Accuracy (%)	
	Mean	Std.	Mean	Std	Mean	Std	Mean	Std.	Mean	Std.	Mean	Std.
LR	78,00	3,19	57,86	8,49	88,80	6,25	73,92	5,80	64,46	6,25	73,33	4,01
RF	77,07	3,92	59,33	6,88	87,71	5,61	70,27	7,24	64,63	7,36	73,52	4,33
ANN	75,91	3,34	60,88	12,60	81,56	9,26	68,33	6,15	66,53	5,31	71,22	5,92
DT	73,31	4,94	51,92	12,02	82,07	8,11	67,62	5,54	58,00	8,07	67,00	4,20
NB	75,27	3,90	48,92	8,71	89,40	6,70	71,40	7,95	57,71	7,76	69,16	4,70
kNN	75,39	3,34	51,52	10,61	88,20	7,65	70,38	6,31	58,76	8,62	69,86	4,69
SVC	77,87	3,37	56,01	11,31	89,60	8,09	74,78	5,53	63,20	8,58	72,81	4,88
NLCv1	**85,13**	2,32	68,08	10,58	94,00	7,26	86,67	7,09	**75,33**	5,60	**81,04**	3,94

Table 11
The classification results of the proposed NLC paradigm against the peer competitors on the Haberman's survival dataset.

	Accuracy (%)		Sensitivity (%)		Specificity (%)		Precision (%)		F1-Score (%)		Balanced Accuracy (%)	
	Mean	Std.	Mean	Std	Mean	Std	Mean	Std.	Mean	Std.	Mean	Std.
LR	74,85	5,37	17,22	11,30	95,59	9,20	61,67	38,94	26,16	16,35	56,41	7,11
RF	72,28	5,25	17,22	12,61	91,72	9,54	42,00	31,37	22,12	15,79	54,47	6,47
ANN	73,89	6,25	20,83	10,70	95,94	9,37	52,62	39,42	26,61	19,45	58,39	8,03
DT	71,91	3,11	17,08	16,50	89,24	10,74	16,42	25,99	13,85	18,04	53,16	4,97
NB	75,18	6,27	20,83	15,48	94,70	12,17	61,17	39,44	29,47	19,77	57,77	8,87
kNN	74,54	5,86	20,83	13,31	93,81	10,54	55,00	38,41	29,23	18,10	57,32	7,77
SVC	71,91	3,74	8,61	5,65	94,72	4,88	45,83	39,66	13,93	9,25	51,67	4,11
NLCv1	**85.33**	2.81	55.00	6.29	96.36	4.69	85.95	5.16	**90.63**	1.49	**75.68**	7.89

Table 12
Results for Cleveland Heart Disease dataset.

	Accuracy (%)		Sensitivity (%)		Specificity (%)		Precision (%)		F1-Score (%)		Balanced Accuracy (%)	
	Mean	Std.	Mean	Std	Mean	Std	Mean	Std.	Mean	Std.	Mean	Std.
LR	81,53	6,93	87,23	9,77	74,59	8,40	80,94	7,11	83,61	6,61	80,91	7,02
RF	82,85	4,71	89,67	8,93	74,69	7,12	82,11	6,23	84,58	4,97	82,18	5,30
ANN	79,21	5,81	86,70	9,48	75,83	6,48	82,19	6,57	79,95	6,86	81,27	3,49
DT	80,55	6,26	81,76	5,93	71,30	4,66	80,55	9,03	76,29	6,43	76,53	3,39
NB	83,53	6,10	92,05	8,50	73,12	7,46	81,08	6,89	85,86	5,42	82,59	6,43
kNN	83,52	6,10	87,26	8,32	78,96	7,27	83,79	6,85	85,18	5,65	83,11	6,22
SVC	83,84	5,93	91,52	8,31	74,59	7,17	81,60	6,13	85,99	5,37	83,05	6,03
NLCv1	**86.15**	4.20	90.62	4.17	80.88	4.39	85.40	6.58	**87.71**	3.16	**85.75**	4.62

Table 13
Results for Mammographic Mass dataset.

	Accuracy (%)		Sensitivity (%)		Specificity (%)		Precision (%)		F1-Score (%)		Balanced Accuracy (%)	
	Mean	Std.	Mean	Std	Mean	Std	Mean	Std.	Mean	Std.	Mean	Std.
LR	80,39	4,56	84,34	7,66	76,64	6,11	77,48	4,80	80,56	4,85	80,49	4,56
RF	80,39	5,34	85,34	9,58	75,70	7,65	76,90	4,56	80,72	6,04	80,52	5,73
ANN	80,14	4,88	85,07	9,03	76,85	7,16	76,64	4,35	80,84	5,57	80,96	5,30
DT	79,91	5,59	86,08	7,65	72,33	6,19	77,59	5,74	79,23	7,61	79,20	4,73
NB	79,30	4,42	85,83	6,71	73,14	5,57	75,13	4,34	80,02	4,57	79,48	4,42
kNN	77,98	7,72	77,87	9,87	78,06	8,81	77,00	7,92	77,28	8,35	77,97	7,75
SVC	79,66	3,82	86,57	7,50	73,13	5,65	75,41	4,07	80,39	4,11	79,85	3,81
NLCv1	**81.46**	4.10	86.25	5.03	76.90	4.52	78.66	6.30	**82.03**	3.35	**81.58**	4.00

Table 14
Results for NBA Rookie Longevity dataset.

	Accuracy (%)		Sensitivity (%)		Specificity (%)		Precision (%)		F1-Score (%)		Balanced Accuracy (%)	
	Mean	Std.	Mean	Std	Mean	Std	Mean	Std.	Mean	Std.	Mean	Std.
LR	71,19	4,32	82,21	5,25	53,08	4,96	74,32	3,62	77,97	3,42	67,65	4,68
RF	70,06	3,66	79,67	5,59	55,65	4,98	73,96	2,67	77,21	3,53	67,66	4,37
ANN	71,11	4,39	80,40	4,66	53,99	4,86	75,61	2,84	76,27	4,34	67,19	5,05
DT	67,95	4,23	76,41	7,40	54,24	5,05	73,47	3,87	73,53	3,53	65,33	2,71
NB	61,55	4,96	48,43	7,51	83,11	5,90	82,30	4,45	60,74	6,66	65,77	4,29
kNN	69,23	3,06	78,58	5,53	53,88	4,17	73,71	2,13	75,97	2,95	66,23	2,81
SVC	71,03	4,28	83,55	5,51	50,50	5,16	73,64	3,75	78,16	3,33	67,02	4,81
NLCv1	**72.45**	1.90	86.84	6.01	48.81	4.79	73.92	3.55	**79.64**	1.39	**67.82**	3.56

This network also achieved superior accuracy and outperformed all peer competitors with an F1-Score of 87.80% and balanced accuracy of 87.85%.

In the last benchmark, NBA Rookie Longevity dataset, NLC built a very sparse network using three attributes (***GP₀***, ***OREB₂***, and ***TOV₀***) as inputs and achieved 77.03% accuracy on the test set of a cross-validation fold, with an F1-score of 81.91% and balanced accuracy of 74.88% (see Fig. 12).

6. Conclusion and future work

Unlike today's artificial neurons, biological neurons fire in discrete pulses. Although the original concept of McCulloch and Pitts was based on a calculus of ***logic***, ANNs have been transformed from being a mechanism that imitates the functioning of the human brain and turned into an ***analog*** computational model to solve advanced classification problems. In addition to being biologically implausible, the major drawback of recent ANN models is the vast necessity of computation power to process data. It sometimes takes days and weeks to train a deep learning system using a multitude of graphical processing units (GPU) to classify tiny low-resolution images or learn a basic atari game to outperform humans. Today, only a handful of researchers under the hood of tech giants like Google and Facebook are able to use resources to experiment with relatively bigger datasets and produce superior models. Even after decades of significant progress, it is virtually impossible for ANNs to develop a high-performing system which can perceive the real world in full resolution and demonstrate general intelligence like humans. Therefore, further research on generic Artificial Neural Networks – no matter how deep they are – is unsustainable.

Back in history, John Von Neumann laid the foundations of today's digital computers, profoundly inspired by the landmark paper by McCulloch and Pitts [9], who applied Boolean algebra to the behavior of the nervous system by explaining that the excitatory or inhibitory nerve pulses are equivalent to logical propositions. Von Neumann regarded the nervous system as ***prima facie digital*** and argued that stimulation of nerve pulses by appropriate combinations of other nerve pulses makes the neuron comparable to a ***logic gate.*** Depending on its threshold, a biological neuron can perform logical operations of conjunction or disjunction. Von Neumann concluded that the *all-or-none* character of biological neurons can be imitated by telegraph relays or vacuum tubes, which are replaced by transistors of today's high-speed microprocessors.

Research in the field of Neuroscience showed that our digital brain has the remarkable capability of *neuroplasticity*, which enables functional and structural reorganization through a lifetime by rewiring its synaptic connections, generating new pathways, and pruning existing ones in response to internal or external stimuli, as well as a learning activity. Nerve cells are generated or eliminated, even during adulthood. A human brain neuron is

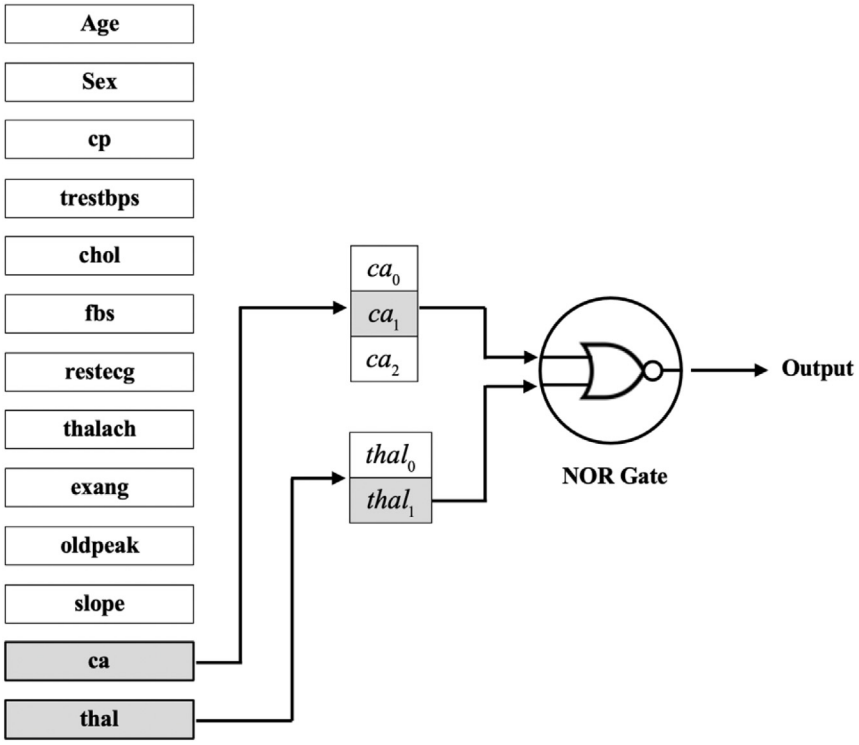


Fig. 10. The resulting NLC Network on the Cleveland Heart Disease dataset. The network has one neural unit (a NOR gate) and two connections. The only neural unit takes inputs from the second binary digit of ***ca*** and the second binary digit of ***thal*** attributes. The network achieved 91.89% accuracy on the test set of a cross-validation fold, with an F1-score of 92.68% and balanced accuracy of 91.62%.

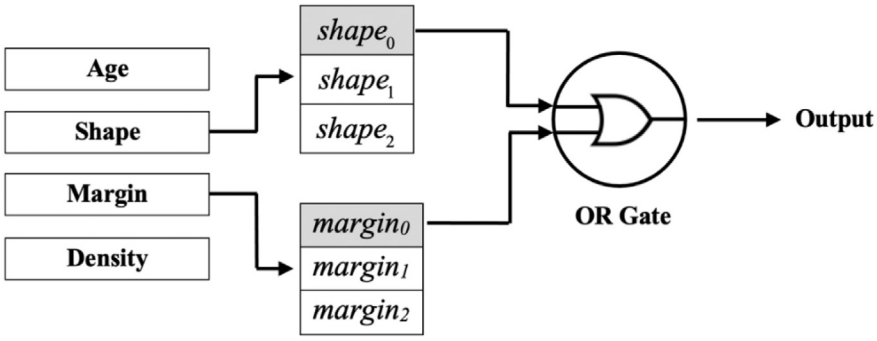


Fig. 11. The resulting NLC Network on Mammographic Mass dataset. The network has one neural unit (an OR gate) and two connections. The only neural unit takes its inputs from the first binary digit of ***shape*** and the first binary digit of ***margin*** attributes. The network achieved 87.80% accuracy on the test set of a cross-validation fold, with an F1-score of 87.80% and balanced accuracy of 87.85%.

thought to be connected to an average of a ***thousand*** other neurons, in an arbitrary manner determined by a three-dimensional arrangement. Although the exact mechanism of how these connections are formed remains elusive, significant evidence suggests that neural rewiring is optimized through the evolution of our brain, which is constantly learning and adapting to its surroundings.

Recent developments in Artificial Neural Networks reveal the importance of how the neurons are laid out and connected since the success of an ANN depends mainly on its architecture. Latest approaches such as ***Neuroevolution*** and ***Reinforcement Learning*** in Neural Architecture Search (NAS) enabled the optimization of network architectures automatically and made a significant impact on the performance of the models by also removing the burden of designing structures through trial and error. Superior models designed with both manual and automatic methods imply that sparse architectures are proved to be efficient since they are biologically more plausible and behave more like the brain. While conventional pruning methods exist for some time, new bright ideas such as ***dropout*** and ***skip connections*** helped achieve state-of-the-art results on image classification tasks.

In this initial paper, we introduced Neural Logic Circuits (NLC) as a new AI paradigm, drawing inspiration from the *Neuroplasticity* of the human brain and the digital character of biological neurons. The NLC can be defined as a weightless neural network that learns by the evolution of its architecture through the rewiring of synaptic connections and generation of artificial neurons functioning as logic gates. The neurons in this structure mimic their biological counterparts by receiving sensory input signals and producing inhibitory or excitatory pulses in an "all-or-none" fashion. Unlike Artificial Neural Networks, NLC is a weightless structure, working with binary data and has no training procedure such as backpropagation. We utilized Genetic Algorithms to design the network structure and optimize itself through generations by using training data. NLC starts from a minimal structure and gradually complexifies its neural circuitry

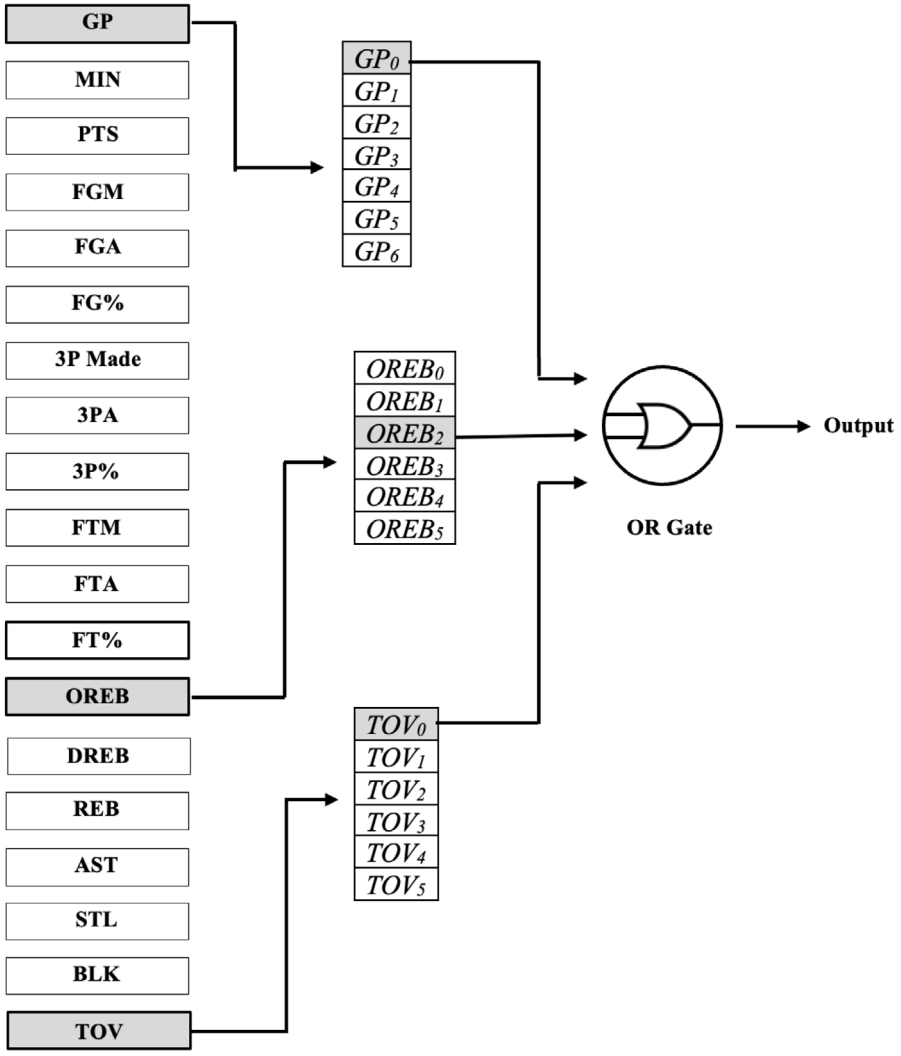


Fig. 12. The resulting NLC Network on the NBA Rookie Longevity dataset. The network has one neural unit (an OR gate) and three connections. The only neural unit takes its inputs from the first binary digit of ***GP (games played),*** the third binary digit of ***OREB (offensive rebounds)*** and the first binary digit of ***TOV (turnovers)*** attributes. The network achieved 77.03% accuracy on the test set of a cross-validation fold, with an F1-score of 81.91% and balanced accuracy of 74.88%.

by adding new neurons and introducing new connections. Finally, the generalization performance is assessed by introducing test instances that the network has never encountered before.

We evaluated our model on well-known binary classification datasets using advanced performance metrics and compared the results with a selection of modern state-of-the-art machine learning algorithms (including the ANNs) as peer competitors. For a fair comparison, we optimized hyperparameters of all models and employed a *Stratified Cross-Validation* procedure to remove any bias which may be caused by train-test splits. For transparency of results, we made all source codes available and ready to reproduce on GitHub. We open-sourced our proposed NLC paradigm and served the complete algorithm to public access.

Extensive experimental data revealed that our initial model, called NLCv1 outperformed all peer competitors in all test cases, yielding encouraging findings for implementing this novel paradigm. Its outstanding performance on F1-Score and Balanced Accuracy shows that NLC is superior in predicting not only in general but also at the decision class of each benchmark. This is extremely important when choosing a model with an asymmetric class distribution, especially in medical data where predicting minority class is crucial (benign-malignant). Furthermore, the networks built by NLC turned out to be sparse architectures, including a few gates and connections. Although the algorithm was programmed to search for larger structures in an augmenting complexity, better results were achieved on smaller networks using one or two attributes for each dataset.

This was the initial version of NLC, and we already know that there is much room for improvement. For future work, we plan to make NLC compatible with all challenging AI tasks, including multi-class classification, regression, image classification, and natural language processing. The paradigm can further be used to be merged with Convolutional Neural Networks (CNN), where the final feature vector provided by CNN is converted and used as binary input to NLC. Besides, several other optimization techniques rather than Genetic Algorithms can be experimented with to optimize the neural circuitry of NLC, such as swarm-based optimization methods or multi-objective optimization algorithms.

CRediT authorship contribution statement

Hamit Taner Ünal: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Visualization. **Fatih Başçiftçi:** Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We would like to express gratitude to our families for their patience and continuous support. We appreciate and express profound thanks to the Library of Selçuk University and publishers worldwide for providing access to the various databases of scientific research.

Patents

This research is protected by international patent laws and treaty provisions with patent application number 2021/020448. Source codes for Neural Logic Circuits (NLC) are available under an open-source license at https://github.com/htanerunal?tab=repositories. Researchers are kindly requested to credit authors with proper attribution and indicate if changes were made.

References

[1] M. Costandi, Neuroplasticity, MIt Press, 2016.
[2] M.A. Hofman, Evolution of the human brain: when bigger is better, Front. Neuroanat. 8 (2014) 15.
[3] A.S. Householder, A theory of steady-state activity in nerve-fiber networks: I. Definitions and preliminary lemmas, Bull. Math. Biophys. 3 (2) (1941) 63–69.
[4] N. Rashevsky, Outline of a physico-mathematical theory of excitation and inhibition, Protoplasma 20 (1) (1933) 42–56.
[5] N. Rashevsky, Some physico-mathematical aspects of nerve conduction, Physics 4 (9) (1933) 341–349.
[6] N. Rashevsky, Outline of a physico-mathematical theory of the brain, J. Gen. Psychol. 13 (1) (1935) 82–112.
[7] M.S. Thomas, J.L. McClelland, Connectionist models of cognition, in: The Cambridge Handbook of Computational Psychology, Cambridge University Press, 2008, pp. 23–58.
[8] A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proc. Lond. Math. Soc. 2 (1) (1937) 230–265.
[9] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[10] A. Goldental, S. Guberman, R. Vardi, I. Kanter, A computational paradigm for dynamic logic-gates in neuronal activity, Front. Comput. Neurosci. 8 (2014) 52.
[11] D. Hebb, Organization of Behavior, Wiley, New York, 1949.
[12] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychol. Rev. 65 (6) (1958) 386.
[13] D. Rumelhart, G. Hinton, R. Williams, Learning Internal Representation by Error Backpropagation, Parallel Distributed Processing: Explorations Microstructure of Cognition, MIT Press, Cambridge, 1986.
[14] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, 1985, Retrieved from.
[15] J.L.G. Rosa, Biologically plausible artificial neural networks, in: Artificial Neural Networks-Architectures and Applications, IntechOpen, 2013.
[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Paper Presented at the 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009.
[17] R. Chrisley, S. Begeer, Artificial Intelligence: Critical Concepts, Vol. 1, Taylor & Francis, 2000.
[18] L.A. Jeffress, Cerebral Mechanisms in Behavior; The Hixon Symposium, 1951.
[19] J. Von Neumann, The Computer and the Brain, Yale University Press, New Haven, 1958.
[20] S.R.y. Cajal, Textura del Sistema Nervioso del Hombre y de los Vertebrados, Imprenta y Librería de Nicolás Moya, Madrid, 1904, pp. 1899–1904.
[21] C. Golgi, The neuron doctrine: theory and facts, in: Nobel Lecture, Vol. 1921, 1906, pp. 190–217.
[22] G.M. Shepherd, Foundations of the Neuron Doctrine, Oxford University Press, 2015.
[23] C.S. Sherrington, Observations on the scratch-reflex in the spinal dog, J. Physiol. 34 (1–2) (1906) 1–50.
[24] P. Mateos-Aparicio, A. Rodríguez-Moreno, The impact of studying brain plasticity, Front. Cell. Neurosci. 13 (66) (2019).
[25] E. Tanzi, Facts and inductions in current histology of the nervous system, in: Rivista Sperimentale di Freniatria e Medicina Legale delle Mentali Alienazioni, Vol. 19, 1893, pp. 419–472.
[26] J. Konorski, Conditioned reflexes and neuron organization, 1948.
[27] G. Piccinini, The first computational theory of mind and brain: a close look at mcculloch and pitts's logical calculus of ideas immanent in nervous activity, Synthese 141 (2) (2004) 175–215.
[28] P. Cull, General two factor models, Bull. Math. Biophys. 29 (2) (1967) 405.
[29] P. Cull, The mathematical biophysics of nicolas rashevsky, BioSystems 88 (3) (2007) 178–184.
[30] J.D. Cowan, Von Neumann and neural networks, Legacy John Von Neumann 50 (50) (1990) 243.
[31] A. Gefter, The man who tried to redeem the world with logic, Nautilus 21 (2015).
[32] R.M. Golden, Artificial neural networks, in: The Handbook of Technology Management, Vol. 3, Wiley, New York, 2010, pp. 331–346.
[33] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314.
[34] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Netw. 2 (3) (1989) 183–192.
[35] R. Hecht-Nielsen, Kolmogorov's mapping neural network existence theorem, in: Paper Presented at the Proceedings of the International Conference on Neural Networks, 1987.
[36] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[37] A.N. Kolmogorov, On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition, in: Paper Presented at the Doklady Akademii Nauk, 1957.
[38] K.J. Hole, S. Ahmad, A thousand brains: toward biologically constrained AI, SN Appl. Sci. 3 (8) (2021) 1–14.
[39] S.M. Bohte, J.N. Kok, H. La Poutre, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (1–4) (2002) 17–37.
[40] J. Vreeken, Spiking Neural Networks, An Introduction, Adaptive Intelligence Laboratory, 2003.
[41] J.R. Pelàez, J.R. Castillo Piqueira, Biological clues for up-to-date artificial neurons, in: Computational Intelligence, Springer, Boston, MA, 2007, pp. 131–146.
[42] J.C. Whittington, R. Bogacz, Theories of error back-propagation in the brain, Trends in Cognitive Sciences 23 (3) (2019) 235–250.
[43] F. Crick, The recent excitement about neural networks, Nature 337 (6203) (1989) 129–132.
[44] R.C. O'Reilly, Six principles for biologically based computational models of cortical cognition, Trends in Cognitive Sciences 2 (11) (1998) 455–462.
[45] F. Pulvermüller, R. Tomasello, M.R. Henningsen-Schomers, T. Wennekers, Biological constraints on neural network models of cognitive function, Nat. Rev. Neurosci. (2021) 1–15.
[46] Y. LeCun, J.S. Denker, S.A. Solla, Optimal brain damage, in: Paper Presented at the Advances in Neural Information Processing Systems, 1990.
[47] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, 2012, arXiv preprint arXiv:1207.0580.
[48] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Paper Presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
[49] R.K. Srivastava, K. Greff, J. Schmidhuber, Highway networks, 2015, arXiv preprint arXiv:1505.00387.
[50] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, S. Yang, Adanet: Adaptive structural learning of artificial neural networks, in: Paper Presented at the International Conference on Machine Learning, 2017.
[51] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Paper Presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[52] Z. Xu, Z. Wang, J. Li, T. Jin, X. Meng, S. Gao, Dendritic neuron model trained by information feedback-enhanced differential evolution algorithm for classification, Knowl.-Based Syst. 233 (2021) 107536.
[53] S. Ghosh-Dastidar, H. Adeli, Spiking neural networks, Int. J. Neural Syst. 19 (04) (2009) 295–308.
[54] S.R. Kheradpisheh, T. Masquelier, Temporal backpropagation for spiking neural networks with one spike per neuron, Int. J. Neural Syst. 30 (06) (2020) 2050027.
[55] X. Wang, X. Lin, X. Dang, Supervised learning in spiking neural networks: A review of algorithms and evaluations, Neural Netw. 125 (2020) 258–280.
[56] S. Chavlis, P. Poirazi, Drawing inspiration from biological dendrites to empower artificial neural networks, Curr. Opin. Neurobiol. 70 (2021) 1–10.

[57] C. Koch, T. Poggio, V. Torre, Nonlinear interactions in a dendritic tree: localization, timing, and role in information processing, Proc. Natl. Acad. Sci. 80 (9) (1983) 2799–2802.
[58] A. Vandesompele, F. Wyffels, J. Dambre, Dendritic computation in a point neuron model, in: Paper Presented at the International Conference on Artificial Neural Networks, 2020.
[59] H. Agmon-Snir, C.E. Carr, J. Rinzel, The role of dendrites in auditory coincidence detection, Nature 393 (6682) (1998) 268–272.
[60] J. Ji, C. Tang, J. Zhao, Z. Tang, Y. Todo, A survey on dendritic neuron model: Mechanisms, algorithms and practical applications, Neurocomputing (2022).
[61] Z. Song, Y. Tang, J. Ji, Y. Todo, Evaluating a dendritic neuron model for wind speed forecasting, Knowl.-Based Syst. 201 (2020) 106052.
[62] E. Filho, A. de Carvalho, Evolutionary design of MLP neural network architectures, in: Paper Presented at the Proceedings of the 4th Brazilian Symposium on Neural Networks (SBRN'97), 1997.
[63] B. Fullmer, R. Miikkulainen, Using marker-based genetic encoding of neural networks to evolve finite-state behaviour, in: Paper Presented at the Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, 1992.
[64] F. Gruau, Cellular encoding as a graph grammar, in: Paper Presented at the IEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives, 1993.
[65] H. Kitano, Designing neural networks using genetic algorithms with graph generation system, Complex Syst. 4 (4) (1990) 461–476.
[66] S. Luke, L. Spector, Evolving graphs and networks with edge encoding: Preliminary report, in: Paper Presented at the Late Breaking Papers at the Genetic Programming 1996 Conference, 1996.
[67] G. Weiß, Towards the synthesis of neural and evolutionary learning, in: Progress in Neural Networks, Vol. 5, Ablex Publishing Corporation, 1997, pp. 145–176.
[68] G.F. Miller, P.M. Todd, S.U. Hegde, Designing neural networks using genetic algorithms, in: Paper Presented at the ICGA, 1989.
[69] M. Karnaugh, The map method for synthesis of combinational logic circuits, Trans. Am. Inst. Electr. Eng. I 72 (5) (1953) 593–599.
[70] E.J. McCluskey, Minimization of Boolean functions, Bell Syst. Tech. J. 35 (6) (1956) 1417–1444.
[71] W.V. Quine, A way to simplify truth functions, Amer. Math. Monthly 62 (9) (1955) 627–631.
[72] R.K. Brayton, G.D. Hachtel, C. McMullen, A. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis, Vol. 2, Springer Science & Business Media, 1984.
[73] R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, A.R. Wang, MIS: A multiple-level logic optimization system, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 6 (6) (1987) 1062–1081.
[74] C.A. Coello, A.D. Christiansen, A.H. Aguirre, Using genetic algorithms to design combinational logic circuits, in: Intelligent Engineering through Artificial Neural Networks, Vol. 6, 1996, pp. 391–396.
[75] C.A. Coello, A.D. Christiansen, A.H. Aguirre, Automated design of combinational logic circuits using genetic algorithms, in: Paper Presented at the Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, 1997.
[76] H.-K. Lam, F.H. Leung, Design and training for combinational neural-logic systems, IEEE Trans. Ind. Electron. 54 (1) (2007) 612–619.
[77] S.J. Louis, G.J. Rawlins, Designer genetic algorithms: Genetic algorithms in structure design, in: Paper Presented at the ICGA, 1991.
[78] C. Reis, J.T. Machado, J.B. Cunha, Evolutionary design of combinational logic circuits, J. Adv. Comput. Intell. 5 (2004) 7.
[79] J. Holland, Adaptation in natural and artificial systems: an introductory analysis with application to biology, in: Control and Artificial Intelligence, 1975.
[80] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (2) (1988) 95–99.
[81] M. Arslan, M. Çunkaş, T. Sağ, Determination of induction motor parameters with differential evolution algorithm, Neural Comput. Appl. 21 (8) (2012) 1995–2004.
[82] P. Koehn, Combining Genetic Algorithms and Neural Networks: The Encoding Problem, 1994.
[83] D.E. Goldberg, GAs in Search, Optimization and Machine Learning, Edition Addison Wesley Publishing Company, 1989.
[84] B.L. Happel, J.M. Murre, Design and evolution of modular neural network architectures, Neural Netw. 7 (6–7) (1994) 985–1004.
[85] P. Reed, B.S. Minsker, D.E. Goldberg, A multiobjective approach to cost effective long-term groundwater monitoring using an elitist nondominated sorted genetic algorithm with historical data, J. Hydroinform. 3 (2) (2001) 71–89.
[86] P.M. Reed, B.S. Minsker, D.E. Goldberg, The practitioner's role in competent search and optimization using genetic algorithms, in: Paper Presented at the Bridging the Gap: Meeting the World's Water and Environmental Resources Challenges, 2001.
[87] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evol. Comput. 8 (2) (2000) 173–195.
[88] D. Dumitrescu, B. Lazzerini, L. Jain, A. Dumitrescu, International Series of Computational Intelligence. Evolutionary Computation, CRC Press, Boca Raton, 2000.
[89] I.J. Leno, S.S. Sankar, M.V. Raj, S. Ponnambalam, An elitist strategy genetic algorithm for integrated layout design, Int. J. Adv. Manuf. Technol. 66 (9–12) (2013) 1573–1589.
[90] K.O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, Evol. Comput. 10 (2) (2001) 99–127.
[91] J.W. Smith, J.E. Everhart, W. Dickson, W.C. Knowler, R.S. Johannes, Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, in: Paper Presented at the Proceedings of the Annual Symposium on Computer Application in Medical Care, 1988.
[92] T. Sağ, H. Kahramanli, Classification rule mining approach based on multiobjective optimization, in: Paper Presented at the 2017 International Artificial Intelligence and Data Processing Symposium, IDAP, 2017.
[93] K.M. Almustafa, Prediction of heart disease and classifiers' sensitivity analysis, BMC Bioinformatics 21 (1) (2020) 1–18.
[94] M. Elter, R. Schulz-Wendtland, T. Wittenberg, The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process, Med. Phys. 34 (11) (2007) 4164–4172.