Mathematics

AIMS Mathematics, 10(1): 159–194.
DOI: 10.3934/math.2025009
Received: 07 August 2024
Revised: 25 November 2024
Accepted: 27 November 2024
Published: 03 January 2025

https://www.aimspress.com/journal/Math

Research article

Higher order Weighted Random k Satisfiability ($k = 1,3$) in Discrete Hopfield Neural Network

Xiaoyan Liu[1,2], Mohd Shareduwan Mohd Kasihmuddin[2,*], Nur Ezlin Zamri[3], Yunjie Chang[2,4], Suad Abdeen[2] and Yuan Gao[2,5]

[1] School of General Education, Guangzhou College of Technology and Business, Guangzhou 510850, China
[2] School of Mathematical Sciences, Universiti Sains Malaysia, Penang 11800 USM, Malaysia
[3] Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor, Malaysia
[4] School of Computer Science and Engineering, Hunan Institute of Technology, 421002 Hengyang, China
[5] School of Medical Information Engineering, Chengdu University of Traditional Chinese Medicine, 610037 Chengdu, China

* **Correspondence:** Email: shareduwan@usm.my; Tel: +60149442519.

Abstract: Researchers have explored various non-systematic satisfiability approaches to enhance the interpretability of Discrete Hopfield Neural Networks. A flexible framework for non-systematic satisfiability has been developed to investigate diverse logical structures across dimensions and has improved the lack of neuron variation. However, the logic phase of this approach tends to overlook the distribution and characteristics of literal states, and the ratio of negative literals has not been mentioned with higher-order clauses. In this paper, we propose a new non-systematic logic named Weighted Random k Satisfiability ($k = 1,3$), which implements the ratio of negative literals in higher-order clauses. The proposed logic, integrated into the Discrete Hopfield Neural Network, established a logical structure by incorporating the ratio of negative literals during the logic phase. This enhancement increased the network's storage capacity, improving its ability to handle complex, high-dimensional problems. The advanced logic was evaluated in the learning phase by various metrics. When the values of the ratio were $r = 0.2$, 0.4, 0.6, and 0.8, the logic demonstrated the potential for better performances and smaller errors. Furthermore, the performance of the proposed logical structure demonstrated a positive impact on the management of synaptic weights. The results indicated that the optimal global minimum solutions are achieved when the ratio of negative literals was set to $r = 0.8$. Compared to the state-of-the-art logical structures, this novel approach has a more significant impact on achieving global minimum solutions, particularly in terms of the ratio of negative literals.

Keywords: Weighted Random k Satisfiability; non-systematic satisfiability; Discrete Hopfield Neural Network; optimization problem; global solution
Mathematics Subject Classification: 37M05, 37M22, 68N17

1. Introduction

Artificial intelligence (AI) refers to the simulation or approximation of human intelligence by software-coded heuristics. Artificial neural networks (ANNs), inspired by the intricate workings of the human brain, have sparked a worldwide revolution in artificial intelligence [1]. ANNs has become a popular standard approach to solve the optimization problems about several ranges of real-life problems such as in tendency forecasting of economic growth [2], the definition of "smart city" based on transportation [3], the distance education of university students [4], and in the business applications [5]. ANNs have been desired to have the ability to simulate the work that human brains do. The solid structure of the ANN made it limited to showing the real understanding of human brain activities. Although artificial intelligence excels in complex prediction tasks and most AI researchers rely on the ability of the ANN, it is limited in many areas, often being referred to as black-box models [6]. As computers can do complex work faster than humans and can store information that can help to make final decisions, Hopfield and Tank first added a simple variant of ANN into the Hopfield neural network (HNN) in 1985 [7]. The HNN consists of interconnected neurons without hidden neurons. Each neuron is connected through synaptic weight. HNN has the capability to access memory address which is called Content addressable memory (CAM). CAM is a special type of memory that can retrieve specific data through input content, helping network devices quickly retrieve specific data and improving network performance [8,9]. The HNN has two structures which are namely discrete and continuous [10]. Furthermore, as highlighted in other research, the Discrete Hopfield Neural Network (DHNN) has been a central focus due to its rapid development and wide range of applications. Abdullah (1992) added the logic structure named Satisfiability (SAT) into the ANN that leads DHNN to attain better performances obtaining the synaptic weight during the training phase [11]. SAT plays an important role in HNN with the storage capacity of the meaningful inspects of information. This unique structure of logic has the capability of transforming various problems into mathematical formulations. By utilizing special neurons that process a large amount of input information, the results from different aspects of the logic can help explain the predicted outputs. SAT is a powerful tool in automatic reasoning, enabling complex problem-solving across various domains in computer science and artificial intelligence.

SAT refers to determining whether there exists an assignment of variables to a Boolean formula that makes the formula true. SAT is of significant importance in computer science, both as a fundamental topic in theoretical computational complexity research and for its wide-ranging practical applications in fields, such as circuit design [12], software verification [13], artificial intelligence [14,15], and compiler optimization [16]. One of the earliest works by Abdullah proposed Horn satisfiability (Horn SAT) as a new concept of SAT [11]. Horn SAT has been considered a good representation in the field of satisfiability [17]. The behavior of Horn SAT allows the cost function to be zero during the learning phase and decides the dynamic of the neuron state of HNN. Despite the results being able to locate more than 90% of global minimum solutions, the properties of Horn SAT seem fixed. The results demonstrate that DHNN can be implemented by this kind of logic rule. Since then, several researchers have conducted a significant amount of outstanding research work on the improvement of the structure of logic. Kasihmuddin et al. [18] introduced a systematic logic rule named 2-Satisfiability (2SAT) which consists of strictly two literals per clause. Furthermore, a training algorithm was implemented in the training phase of DHNN which can help to minimize the cost function and can also obtain the optimal synaptic weight. Thereafter, the search for this kind of optimal SAT has come a long way. Zhu et al. [19] extended the order of the logic into 3-Satisfiability (3SAT) which proposed a higher-ordered SAT. All the clauses are expressed in Disjunctive Normal Form (DNF), with each literal connected through Conjunctive Normal Form (CNF) within every clause. This study observes that, despite the use of optimized algorithms, the solution space grows exponentially as the size of 3SAT instances increases. Therefore, solving large 3SAT instances demands significant energy consumption on traditional serial electronic computers. Mansor et al. [20] discussed various metrics to justify the capabilities of 3SAT. Moreover, they found that when the order of a clause exceeds 3, it can be reduced to a combination of clauses with order 3 or lower, according to the reduction theorem. The logical rule was embedded into DHNN by comparing the cost function with the Lyapunov energy function. The synaptic weights are derived from the comparison of these two functions. When the number of neurons increased, the synaptic weight values seemed suboptimal and the number of global minimal solutions also decreased. Due to the inflexibility of the logical rules, DHNN always lacks universality.

Addressing the need for varying logic clause structures during the retrieval process, Mouhoub et al. [21] proposed a non-systematic satisfiability approach. This method checks whether a solution to an SAT problem remains valid when new clauses are added and can efficiently adjust the solution to satisfy both the original formula and the new clauses if necessary. Sathasivam et al. [22] later established a new logic rule named Random k satisfiability (RANkSAT) where $k = 2$. This work first mentioned the formula combination consists of first-order logic and second-order logic. The clauses are combined using disjunction, while the literals within each clause are connected through conjunction. The main idea of the implementation of RAN*k*SAT is the flexibility to represent the number of literals that are not limited to only one variable per clause [22]. It was proposed by introducing the random structure that involves first and second-order SAT logical rules. This work was later extended using higher orders for non-systematic logical rules. The study of Karim et al. [23], shows different variants of Random 3 Satisfiability (RAN3SAT). The combination structures of all possible clauses were all implemented in DHNN. RAN3SAT has proposed a method with a high variation value, high global minima ratio, and low learning error. It also concluded that the logic structure with $k = 2,3$, which means the combination contains only second-order and third-order logic clauses would reach a higher probability of minimizing the cost function and better management of synaptic weight management, which leads to lower energy. Alway et al. [24] developed another variation of logic structure that used second-order clauses as the domain clause in the situation of non-systematic SAT. This special logic was named Major 2-satisfiability (MAJ2SAT) where the ratio of the second-order clause is larger than the ratio of the third-order clause. Notably, the overfitting occurs when the ratio of second-order clauses increases in MAJ2SAT. This is because when the number of third-order clauses decreases the probability of satisfied interpretation will also decrease. The work also showed that it could have a lower similarity value than the existing systematic logic rules.

In a subsequent development, Guo et al. [25] introduced a novel logic rule named Y-Random 2 satisfiability (YRAN2SAT), which explored stimulation data within non-systematic logic rule structures. This approach merged both systematic and non-systematic characteristics, combining logical rules with symbolic instructions, thereby significantly enhancing the effectiveness of DHNN. Furthermore, Gao et al. [26] introduced a new variation of RAN3SAT named G-Type Random 3-Satisfiability (GRAN3SAT) that capitalized both higher-order systematic and non-systematic logical rules in DHNN. This was the first attempt to add both logical rules into DHNN. In this work, they present all the sets of logical rule structures that have been proposed. The higher-order systematic logical rule provides storage capacity to GRAN3SAT, whereas the higher-order nonsystematic logical rule provides a more diversified third-order logical connection. Despite its wide range of conclusions of logic rules and various metrics discussed in the following work, this work lacks control of the clear distribution of the number of negative literals. Although these fruitful SAT models have yielded effective results in DHNN, there have been fewer outcomes regarding the discussion of the ratio of negative literals in the logic rules. Research in non-systematic logic has advanced the storage capacity of neural networks. However, there has been limited exploration into leveraging higher-order clauses with varying ratios of negative literals. Incorporating these higher-order clauses, which introduce diverse proportions of negative literals, has the potential to significantly broaden the search space and enhance the storage capabilities of neural networks. This represents a promising area for future research that could lead to more efficient and scalable neural network architectures.

To address these challenges mentioned above, our proposed approach will provide a more flexible structure that incorporates first-order and third-order logic which was randomly selected with a given ratio of negative literals in each clause. In the rest of this paper, we propose a newly non-systematic satisfiability logic named Weighted Random k Satisfiability ($k=1,3$) (W_{r3SAT}). The proposed work demonstrates the effectiveness of integrating the unique logical structure. The key contributions of the details are outlined as follows:

(1) To formulate a logic rule named Weighted Random k Satisfiability that proposes a structural feature of non-systematic satisfiability. The establishment of kth-order clauses is set randomly with this logic added the ratio of negative literals to the first-order logic and third-order logic clauses. Exhaustive Search (ES) is used to find the correct logic structure for the right number of negative literals.

(2) Implementing the proposed Weighted Random k Satisfiability in DHNN enables effective network governance. This proposed logical rule stores information externally and is considered satisfied only if all clauses are satisfiable. In the present framework, each variable in the logical rule is represented as a neuron.

(3) To propose the function in the learning phase to find the minimum cost function for each clause that is satisfied. The combination structure of this logic has more variables and can provide more literals in representing neuron states in DHNN. The optimal synaptic weights are determined by comparing the cost function with the Lyapunov energy function.

(4) To evaluate the capacity of the proposed network by simulated metrics. We utilized different metrics to evaluate the logical structure, and mentioned that the proposed logical rule has the capacity to perform well with lower error in the learning phase and retrieval phase.

(5) To compare the impact of the proposed non-systematic logical rule with existing logics in DHNN by employing various performance metrics. Several factors are considered such as the management of synaptic weight, learning errors, testing errors, solution variations, and similarity.

The organization of the paper is outlined as follows: The motivation of this work is elaborated in Section 2. The proposed formula of Weighted Random k Satisfiability is introduced in Section 3. In Section 4, we integrate the right logic structure according to the predetermined ratio of negative literals. In Section 5, we add Weighted Random k Satisfiability into DHNN. Various metrics are used to analyze the effectiveness between the proposed logic and other predetermined logic rules in Section 6. In Section 7, we describe the results and discuss the performance of the proposed logic by different metrics. The conclusion and further work are discussed in Section 8 at the end of this paper.

2. Weighted Random k Satisfiability

SAT is a Boolean algebra logical rule employed to determine if there exists an assignment of variables that makes a given logical formula true. This rule is fundamental in solving problems where the goal is to find an optimal solution or verify the feasibility of a solution within a specific set of constraints. By evaluating various combinations of variable assignments, the SAT can effectively address complex decision-making and optimization challenges. Each clause in satisfiability contains literals with arbitrary numbers. There are two mainstream divisions of SAT logical rules, namely systematic and non-systematic logic rules in DHNN. In terms of the systematic satisfiability logic, there are strictly 2 literals per clause in 2SAT and 3 literals per clause in 3SAT. On the other hand, in terms of non-systematic satisfiability, there are k literals in each clause with $k=1,2,3$. The current study provides a non-systematic logic named Weighted Random k Satisfiability is determined. In order to introduce the novel Weighted Random k Satisfiability in DHNN. The standard formula of Weighted Random k Satisfiability is set as follows:

a) N variables: $x_1, x_2, \cdots\cdots, x_N$, where $x_i \in \{-1,1\}$ has the bipolar values, $i=1,2,\cdots,N$. Each literal has the same opportunity to be positive or negative.

b) Structure combinations of clauses where:

$$Q=\{r_1, r_2, \cdots, r_p\}, \tag{1}$$

$$r_i=\{u,v\}, \tag{2}$$

$$u+3v=N. \tag{3}$$

Here, u is the number of first-order clauses, v is the number of third-order clauses, $i=1,2,\cdots,p$.

c) The general formulation of the proposed logical structure of Weighted Random k Satisfiability (r3SAT) is formulated as follows:

$$W_{r3SAT}=\wedge_{i=1}^{v} Q_i^{(3)} \wedge_{i=1}^{u} Q_i^{(1)}. \tag{4}$$

As shown in Eq (4), $Q_i^{(1)}$ and $Q_i^{(3)}$ are refer to the first-order and third-order clauses, respectively. Each clause is combined with CNF.

d) In each clause, literals are combined with the DNF, such structures are stated as follows:

$$Q_i^{(k)}=\begin{cases} A_i, & k=1 \\ \left(B_i \vee C_i \vee D_i\right), & k=3 \end{cases}. \tag{5}$$

Literals represent bipolar values $\{A_i, \neg A_i\}$, $\{B_i, \neg B_i\}$, $\{C_i, \neg C_i\}$, $\{D_i, \neg D_i\}$, and so on. Where A_i represents positive and $\neg A_i$ represents negative. In r3SAT, the ratio of the total number of negative literals is defined as r. The approach outlined the range of ratio is between 0.1 and 0.9. The step size is separated randomly. The number of negative literals n in weighted r3SAT can be calculated as follows in Eq (6):

$$n=rN. \tag{6}$$

When $r=0$, it seems that there are no negative literals in every clause. This can be called monotone clauses [27]. Additionally, $r=1$ seems that the whole weighted *r*3SAT is comprised of literals which are all negative without positive ones. The number of rN sometimes shows the particularity that it is not in nature number. For example, when $r=0.4$ and $N=9$ shows that $rN=3.6$ is not a natural number. The rounding function is used to deal with this kind of conflict. The rounding function means to get the maximum natural number which is not bigger than the real number it has been. Since rN must be a natural number, the number of negative literals is 3 under the condition below:

$$rN=3.6. \tag{7}$$

It can be introduced as $\left[3.6\right]=3$ in short. Equations (8)–(11) show examples under this condition when the number of negative literals is equal to 3.

$$W_{r3SAT}=A_1\wedge\neg A_2\wedge A_3\wedge\neg A_4\wedge A_5\wedge A_6\wedge A_7\wedge\neg A_8\wedge A_9, \tag{8}$$

$$W_{r3SAT}=\neg A_1\wedge\neg A_2\wedge A_3\wedge A_4\wedge A_5\wedge A_6\wedge\left(B_1\vee\neg C_1\vee D_1\right), \tag{9}$$

$$W_{r3SAT}=\neg A_1\wedge A_2\wedge A_3\wedge\left(\neg B_1\vee C_1\vee D_1\right)\wedge\left(B_2\vee\neg C_2\vee D_2\right), \tag{10}$$

$$W_{r3SAT}=\left(\neg B_1\vee C_1\vee D_1\right)\wedge\left(B_2\vee\neg C_2\vee D_2\right)\wedge\left(B_3\vee C_3\vee\neg D_3\right). \tag{11}$$

As illustrated in the examples, the number of negative literals is consistently three. As outlined in Eq (8), the logical structure is composed entirely of first-order clauses, with negative literals randomly distributed across them. Equation (11) presents a structure consisting entirely of third-order clauses, where negative literals are assigned to each clause but distributed randomly. Additionally, Eqs (9) and (10) involve both first-order and third-order clauses, with negative literals randomly assigned to each type.

The logic structure of weighted *r*3SAT is dependent on the ratio of r and the nature number N. The weight for each formula generation of any possible $Q_i^{(k)}$ for $k=1,3$ is defined by Eqs (12) and (13) as below:

$$\alpha_i^{(1)}=\begin{cases}0, & if \quad A\\ 1, & if \quad \neg A\end{cases}, \tag{12}$$

$$\alpha_i^{(3)}=\begin{cases}0, & if \quad \left(B\vee C\vee D\right)\\ 1, & if \quad \left(\neg B\vee C\vee D\right),\left(B\vee\neg C\vee D\right),\left(B\vee C\vee\neg D\right)\\ 2, & if \quad \left(\neg B\vee\neg C\vee D\right),\left(B\vee\neg C\vee\neg D\right),\left(\neg B\vee C\vee\neg D\right)\\ 3, & if \quad \left(\neg B\vee\neg C\vee\neg D\right)\end{cases}, \tag{13}$$

where $\alpha_i^{(k)}\in\{0,1,2,3\}$ denotes the number of negative literals in the k th-order clauses for $k=1,3$. When the number of negative literals is equal to 0 per clause, the result of $\alpha_i^{(k)}$ is set to be 0. While there exists only one negative literal per clause, the value of $\alpha_i^{(k)}$ is equal to 1. It sounds self-evident that when the number of negative literals is equal to 1, 2, or 3. $\alpha_i^{(k)}$ is equal to the same value of the number. Equation (12) mentions that in the first-order clause, when the number of negative literals is 1, the value of $\alpha_i^{(1)}$ is equal to one. Otherwise, the value is 0. In Eq (13) which shows the clause of third-order, the value of $\alpha_i^{(3)}$ is equal to the same value of the number of negative literals. Subsequently, the total number of negative literals can be calculated by the summaries in Eq (14) as follows:

$$\eta = \sum_{i=1}^{u} \alpha_i^{(1)} + \sum_{i=1}^{v} \alpha_i^{(3)} . \tag{14}$$

Note that the total weight of negative literals η is equivalent to the total number of negative literals present in the weighted *r*3SAT formulation. One of the most important things during this step is to make sure that Eq (15) can be satisfied.

$$\left| rN - \eta \right| = 0 . \tag{15}$$

Building on the established framework of *r*2SAT [28], the Genetic Algorithm (GA) was utilized to identify the optimal logical structure for each possible combination, with the goal of minimizing the fitness function as defined in Eq (15) during the logic phase. In the context of this study, the fitness function is further introduced as a key method for effectively minimizing Eq (15) within the logic phase, offering a systematic approach to refining the logical structure through iterative optimization. The structure of W_{r3SAT} represents first and third-order clauses combined with non-redundant literals. As mentioned before, the step size for the ratio is limited to 0.1 while the range of the ratio is between 0.1 to 0.9. The generation of *r*3SAT is shown in Figure 1. It is worth mentioning that although the correct W_{r3SAT} can be generated, the condition determined by Eq (15) often cannot be satisfied. Additional conditions must be imposed during the logic phase to ensure that a valid solution for this distribution exists. The random literals must be generated based on the value of the input N while the weight of the logic is evaluated according to the value of weight α_i in each cause. ES is used to generate the correct number of literals in each clause with a linear function [29]. The ratio of negative literals is determined by a specified value. After the generation of these literals, they are randomly distributed across the first-order and third-order logic clauses. The detailed process of this generation will be discussed in the following section. After that, the structure of W_{r3SAT} seems established.

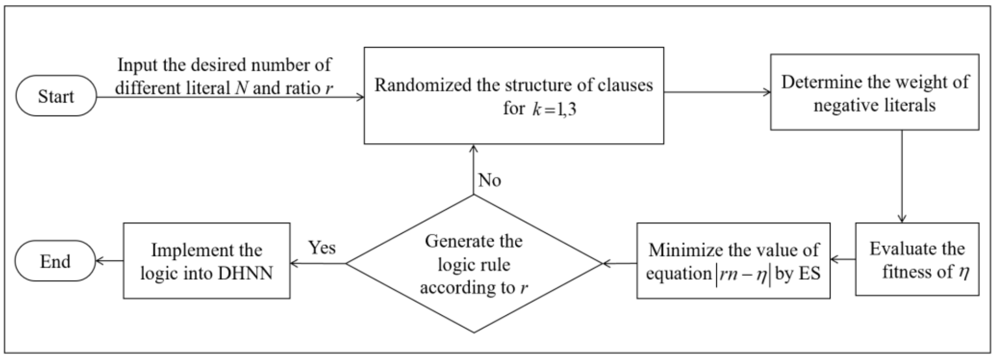

Start
Input the desired number of different literal N and ratio r
Randomized the structure of clauses for k=1,3
Determine the weight of negative literals
Evaluate the fitness of η
Minimize the value of equation |rn − η| by ES
Generate the logic rule according to r
No
Yes
Implement the logic into DHNN
End

Figure 1. Flowchart generating Weighted Random 3 Satisfiability (W_{r3SAT}).

3. Logic generation of Weighted Random 3 Satisfiability

In the following work, ES is employed to address the problem of determining the distribution of negative literals in both first-order and third-order logic. In contrast, the studies in [28,30] utilized Genetic Algorithms (GA) because GA is well-suited for solving continuous optimization problems due to its balanced operators for exploration and exploitation. If GA successfully optimizes the resulting fitness function, the outcomes will generally be at least suboptimal in most cases [30]. GA was employed to propose a structured logic phase, serving as an organized framework for assigning the distribution of negative literals in an unbiased manner [28]. Although GA can handle continuous problems, this usually involves discretizing real variables or using special encoding schemes, which may increase complexity and reduce efficiency [28]. Hassanat A, et al. [31] mentioned that GA usually uses fixed mutation rates and crossover operations, which may show poor adaptability to some problems. ES is an algorithmic strategy that tries all possible solutions in the solution space of a problem one by one to find the optimal solution or a solution that satisfies certain conditions [32]. This approach usually ensures that the global optimal solution is found. The ES method has been used in beam angle optimization [33]. The proposed approach utilizes ES to solve the problem, owing to its reliability in converging toward optimal solutions. To assess the quality of the solutions, a fitness function is employed, which quantifies the number of negative literals. This fitness function is mathematically formulated in Eq (16), providing a clear metric for solution evaluation and guiding the optimization process:

$$f_{W_{r3SAT}} = \left| rN - \eta \right|, \tag{16}$$

when $f_{W_{r3SAT}} = 0$, the fitness function reached the optimal value of $f_{W_{r3SAT}}$. It also illustrates that Eq (15) is satisfied and the correct structure of W_{r3SAT} should be generated. On the other hand, $f_{W_{r3SAT}} \neq 0$ illustrates that the solution of the function is not satisfied, ES failed to find the correct structure of the logic mentioned and it will also fail to obtain the correct number of negative literals.

While GA will repeat from the initialization process when $f_{W_{r3SAT}} \neq 0$ until $f_{W_{r3SAT}} = 0$ is achieved or until a specified number of iterations which has been restricted in advance. The algorithm ES is utilized to get the correct number without duplication testing each possible solution in the solution space of a problem sequentially to identify the optimal solution or a solution that meets specific criteria. For example, in the case where $r = 0.5$ and $N = 9$, while $\left[rN \right] = \left[4.5 \right] = 4$ is a nature number, ES would find a solution satisfies $\eta = 4$ to achieve $f_{W_{r3SAT}} = 0$ in the case of Eq (16). The separation of the value $\alpha_i^{(k)}$ for $k = 1,3$ in each kind of clause should be shown under the condition of absolutely two negative literals in the first-order clause and two negative literals in the third-order clause. Finally, it can be concluded that the possible structure W_{r3SAT} will be generated as follows:

$$W_{r3SAT} = \neg A_1 \wedge A_2 \wedge \neg A_3 \wedge \left(\neg B_1 \vee C_1 \vee D_1 \right) \wedge \left(B_2 \vee \neg C_2 \vee D_2 \right). \tag{17}$$

Equation (17) shows the example of $r = 0.5 \in \left[0.1, 0.9 \right]$. While excessive cases where $r = 0$ or $r = 1$ exist, it indicates that there are no negative literals or that the whole logic is filled with negative literals monotonously. Monotone clauses can result in an inefficient solution to properly managing the synaptic weights. A clause is considered monotone if it contains only unnegated variables or only negated variables [34]. Since a first-order clause has only one satisfied interpretation, its presence has minimal impact on the overall structure of the logic when dealing with negative form literals. The logic phase in Algorithm 1 presents the pseudocode for generating the randomly weighted 3-SAT logic phase.

4. Weighted *r*3SAT in the Discrete Hopfield Neural Network

The effectiveness of DHNN primarily relies on its ability to store information, which is crucial for solving optimization problems. Researchers have demonstrated that this capability can be applied to a wide range of problems due to its versatility in addressing complex issues [24–26]. The fundamental rule for the update of neurons is shown as follows:

$$S_i = \begin{cases} 1, & \sum_{j,k}^{n} W_{ijk}^{(3)} S_j S_k \geq \theta_i \\ -1, & otherwise \end{cases}, \tag{18}$$

where W_{ijk} denotes the synaptic weight between neuron i, neuron j and neuron k. The weight presents the strength of the connections between neurons [26]. S_i denotes the state of neuron i and θ_i is a predetermined threshold value. The property of the synaptic weights for two neurons in the state is always symmetric which has the feature that $W_{ij}^{(2)} = W_{ji}^{(2)}$, $W_{ijk}{}^{(3)}$ have the same value for all permutation combinations with $i \neq j \neq k$, and has no self-connection $W_{ii}^{(2)} = W_{jj}^{(2)} = W_{kk}^{(2)} = 0$, $W_{iii}^{(3)} = W_{jjj}^{(3)} = W_{kkk}^{(3)} = 0$. The predetermined threshold θ_i is always determined as $\theta_i = 0$ which means that the energy in DHNN decreases uniformly [28]. The primary purpose of implementing W_{r3SAT} in DHNN is to reduce logical inconsistency. Our main objective of this work is to minimize the cost function, ideally driving it to zero. This is accomplished by minimizing the network's cost function. The cost function $E_{W_{r3SAT}}$ for the implementation of the logic W_{r3SAT} into DHNN can be formulated as shown in Eq (19) below:

$$E_{W_{r3SAT}} = \sum_{i=1}^{v} \left(\prod_{j=1}^{3} Q_{ij}^{(3)} \right) + \sum_{i=1}^{u} \left(\prod_{j=1}^{1} Q_{ij}^{(1)} \right), \tag{19}$$

where u is the number of first-order clauses and v is the number of third-order clauses. The inconsistency of $Q_{ij}^{(k)}$ ($k = 1,3$) is determined as follows:

$$Q_{ij}^{(1,3)} = \begin{cases} \frac{1}{2}\left(1 + S_{A_i}\right), & if\ A_i \\ \frac{1}{2}\left(1 - S_{A_i}\right), & if\ \neg A_i \end{cases}. \tag{20}$$

Here, S_{A_i} is the neuron state and A_i comprised of bipolar literals $A_i \in \{-1,1\}$. To achieve a minimized cost function that approaches zero, the DHNN must identify at least one interpretation that satisfies the conditions and yields the result of $E_{W_{r3SAT}} = 0$. The formulation of probability (P) of finding the satisfactory interpretation is presented in Eq (21) below [22]. $\left(1-\frac{1}{2^k}\right)$ is the probability of satisfying the k th-order clauses.

$$P(E_{W_{r3SAT}}=0)=\left(1-\frac{1}{2^3}\right)^v\left(1-\frac{1}{2}\right)^u. \tag{21}$$

The value of $E_{W_{r3SAT}}$ depends on the number of unsatisfied clauses in W_{r3SAT}. When the number of unsatisfied clauses increases, the value of $E_{W_{r3SAT}}$ also increases. The minimum value of $E_{W_{r3SAT}}$ indicates the structure is minimized and the optimal values of the synaptic weights are obtained. $E_{W_{r3SAT}}=0$ indicates that all the clauses in this logic are satisfied. It is ensured that the DHNN can converge to the correct final state. According to Wan Abdullah's method (WA) [11], the synaptic weights of the neurons can be determined by comparing the coefficients of the cost function with the Lyapunov energy function of the DHNN. The values of the synaptic weights can be stored in the Content Addressable Memory (CAM). Equation (22) represents the network's local field.

$$H_i=\sum_{k=1,k\neq i,j}^{n}\sum_{j=1,j\neq i}^{n}W_{ijk}^{(3)}S_jS_k+\sum_{j=1,j\neq i}^{n}W_{ij}^{(2)}S_j+W_i^{(1)}. \tag{22}$$

DHNN utilized an activation function of the Hyperbolic Tangent Activation Function (HTAF) to enable the convergence of final neuron states because of its non-linear properties. According to the value of Eq (22), Eq (23) is the generated final neuron states.

$$H_i(t)=\begin{cases}1, & \sum_{k=1,k\neq i,j}^{n}\sum_{j=1,j\neq i}^{n}W_{ijk}^{(3)}S_jS_k+\sum_{j=1,j\neq i}^{n}W_{ij}^{(2)}S_j+W_i^{(1)}\geq 0\\ -1, & otherwise\end{cases}. \tag{23}$$

The initial state and the updated state of the neuron is represented as H_i and $H_i(t)$ as shown in Eqs (22) and (23). $W_i^{(1)}$, $W_{ij}^{(2)}$, and $W_{ijk}^{(3)}$ represent the synaptic weights of the first-order, second-order and third-order in DHNN, respectively. The equation for HTAF is shown in Eq (24) by Mansor [35]:

$$\tanh(H_i)=\frac{e^{H_i}-e^{-H_i}}{e^{H_i}+e^{-H_i}}, \tag{24}$$

and the neuron states are updated by Eq (25) below:

$$S_i(t)=\begin{cases}1, & \tanh(H_i)\geq 0\\ -1, & otherwise\end{cases}. \tag{25}$$

When the evaluation of the cost function primarily depends on the stable state of the neurons, its minimum corresponds to the optimal solution for the given structure. An analysis conducted by Ma et al. [36] demonstrated that the Lyapunov energy is finite and can attain a minimum energy value, further supporting the convergence of the system to an optimal state. The Lyapunov function is formulated as follows:

$$L_W = -\frac{1}{3}\sum_{i=1}^{n}\sum_{k=1,k\neq i,j}^{n}\sum_{j=1,j\neq i}^{n} W_{ijk}^{(3)}S_iS_jS_k - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} W_{ij}^{(2)}S_iS_j - \sum_{i=1}^{n} W_i^{(1)} . \tag{26}$$

After the calculation of all the solution states for the L_W, the result can be concluded that the minimum of all the L_W can be obtained by the number of clauses. The minimum value can be calculated as follows:

$$L_W^{\min} = \frac{1}{8}v + \frac{1}{2}u , \tag{27}$$

where u and v show the number of first-order and third-order logic. Equation (27) demonstrates that finding the minimum value of the Lyapunov function does not require the use of ES; instead, it can be determined based on the number of first- and third-order clauses. This equation provides a simpler method for calculating the minimum value. Furthermore, the optimal synaptic weights can be derived from the coefficients of the functions when $E_{W_{r3SAT}} = 0$ or $\frac{1}{2}(1 \pm S_{A_i}) = 0$ per clause.

Otherwise, the synaptic weights would be obtained randomly. When $L_W^{\min}$ is the expected global minimum energy solution of the DHNN, the final state will also be converged to the lowest energy of the neuron states. The converges of the final neuron state are calculated using Eq (28):

$$\left| L_W - L_W^{\min} \right| \leq Tol , \tag{28}$$

where Tol is a predetermined value of tolerance. Although the smaller the tolerance value the better, it can always be set as $Tol = 0.001$ [23].

Algorithm 1 illustrates the implementation of Weighted Random 3 Satisfiability into DHNN. Equation (28) determines the condition that can be used to detect the behavior of satisfying W_{r3SAT}. Figure 2 illustrates the schematic diagram showing the implementation of W_{r3SAT} in DHNN. The main block represented shows the first-order and third-order clauses after adding the ratio of negative literals. Inside the higher-order logic block, the blue, yellow, and pink lines indicate the connections between the neurons. The diagram consists of the learning phase and the retrieval phase. The clauses are selected randomly during the learning phase for W_{r3SAT} and then converted into Boolean algebra representation.

Algorithm 1: Pseudocode of DHNN-Weighted *r*3SAT.

Input: *Initialize the number of literals N and the ratio of negative literals r ;*
Output: *Global minimum solutions or local minimum solutions*

```
1   Initialization: Generate initial neuron states; ## {Logic phase}
2   Calculate the right number of negative literals by Eq (14)
3   Select all the logic combinations of $W_{r3SAT}$
4   Calculate the number of first and third-order clauses;
5   for $i=1$ to $N$
6       select the optimal solution of $rN$ using ES;
7       if $|rN-\eta|=0$ do
8           calculate the fitness using Eq (16)
9       else
10          return to line 3
11      end if
12  end for
13  store the best solution achieved $f_{W_{r3SAT}}=0$
14  for $i=0$ to $c$ ## {Learning phase}
15      if $E_{W_{r3SAT}}=0$
16          Calculate the synaptic weight using WA method
17          Store synaptic weight in CAM
18      else
19          Generate the synaptic weight randomly
20      end if
21  end for
22  Calculate the global minimum energy $L_W^{\min}$ using Eq (27)
23  Initialize the final neuron state randomly ## {Retrieval phase}
24  for $i=0$ to $c$
25      Calculate the local field by Eq (22)
26      Update the neuron state using HTAF by Eq (24);
27      Calculate the final neuron energy $L_W$ by Eq (26)
28      if $|L_W-L_W^{\min}|\le Tol$
29          Global minimum solutions;
30      else
31          Local minimum solutions;
32      end if
33  end for
```

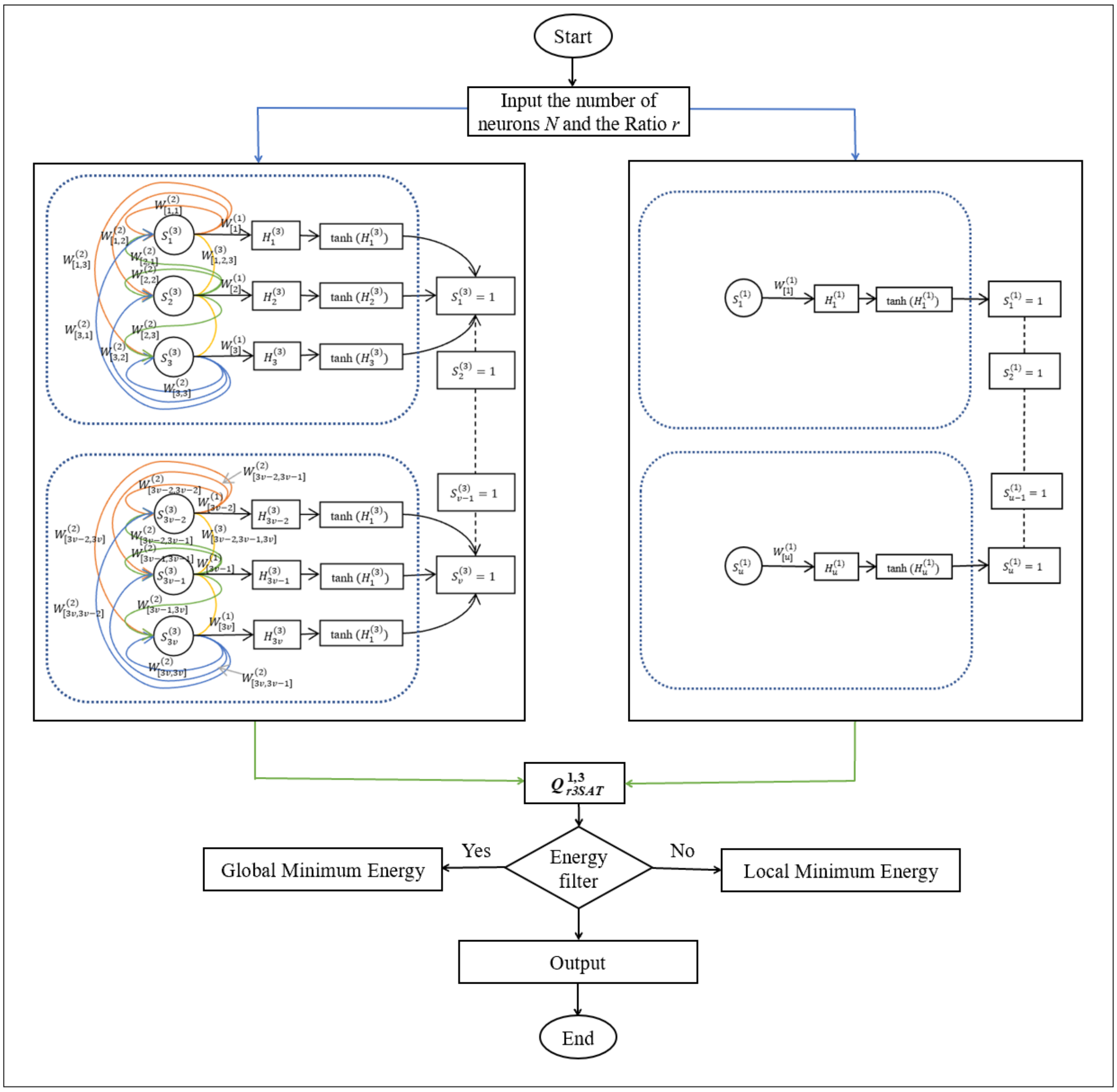


Figure 2. Schematic diagram for DHNN-weighted r3SAT.

Figure 3 demonstrates the flowchart of weighted r3SAT in DHNN. The presentation illustrates how weighted r3SAT is progressed obtaining the synaptic weight in the learning phase. It also indicates the retrieval phase of r3SAT in DHNN. The analysis presented herein confirms the objective of weighted r3SAT is to determine the neuron states that meet the requirements satisfied by the cost function given in Eq (20). Note that optimal global solution is a strong assumption for a general unconstrained optimization problem. However, the primary advantage of using neural networks is their powerful representational capabilities. Therefore, the global minimal solution is a reasonable assumption in DHNN. After selecting the negative literals for each clause, the optimal synaptic weight can be obtained by optimizing the neurons. The acquired synaptic weights are then used to access the desired information during the retrieval phase.

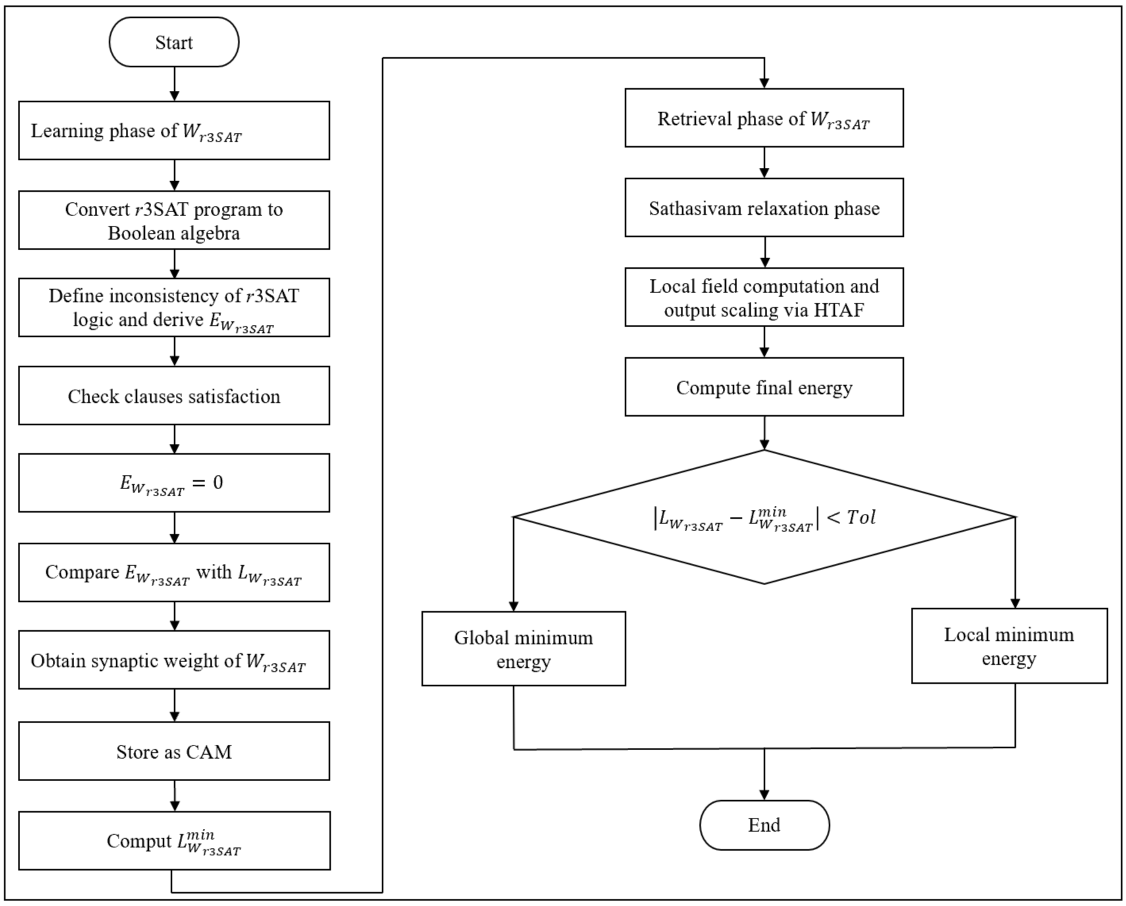

Start
Learning phase of W_{r3SAT}
Convert r3SAT program to Boolean algebra
Define inconsistency of r3SAT logic and derive $E_{W_{r3SAT}}$
Check clauses satisfaction
$E_{W_{r3SAT}} = 0$
Compare $E_{W_{r3SAT}}$ with $L_{W_{r3SAT}}$
Obtain synaptic weight of W_{r3SAT}
Store as CAM
Comput $L_{W_{r3SAT}}^{min}$
Retrieval phase of W_{r3SAT}
Sathasivam relaxation phase
Local field computation and output scaling via HTAF
Compute final energy
$|L_{W_{r3SAT}} - L_{W_{r3SAT}}^{min}| < Tol$
Global minimum energy
Local minimum energy
End

Figure 3. Flowchart of Weighted Random 3SAT in DHNN.

5. Experimental setup

The following work proposed various performance metrics to evaluate the logic output across all phases. The purpose is to judge the effectiveness of adding the ratio into weighted r3SAT.

5.1 Simulation platform

All simulations were conducted using Python 3 on a 64-bit Windows 11 operating system. All experiments were performed on the same computer with the same compiler. The neuron states were generated randomly by the neuron $S_i \in \{-1,1\}$. In the learning phase, the ratios of negative literals are generated by the algorithm ES. Although ES may require significant computational time, it ensures the optimal number of negative neurons. The primary objective of ES is to determine the correct number of negative literals. Third-order clauses in DHNN are crucial for capturing non-linear relationships among variables, thereby extending two-way interactions. These clauses facilitate the handling of complex data patterns, thereby enhancing the accuracy of the models [37]. Several simulations were conducted to evaluate the effectiveness of the proposed satisfiability in both the learning and retrieval phases. Three distinct simulations are discussed in this study, which vary in terms of the number of logic combinations, neurons, and clauses. Additionally, different ratios of negative literals were also tested. Error analysis and neuron similarity analysis are utilized to estimate third-order clause performance in DHNN.

5.2 Parameters assignments

The proposed model will undergo three parts: the logic phase, the learning phase, and the retrieval phase. Each specific detail of the simulations is as below:

a) Different numbers of literals and clauses. The objective of the logic phase is to obtain the right or at least the optimal weighted value of η in the subsequent section. The effectiveness of different logic orders is assessed using various metrics during the learning phase.

b) Various ranges of ratios. Different ratio means different numbers of negative literals which can be used to evaluate the impact on W_{r3SAT}. The performance metrics at every phase can be evaluated and the effects of parameter alterations on W_{r3SAT} can be determined.

c) Different logic structures. The proposed approach shows several existing well-known logic rules that are used to compare with W_{r3SAT} in terms of the flexibility analysis of logic rules. Comparisons were performed using different ranges of neuron numbers as well as different metrics for the mentioned logic rules.

In the logic phase, ES is implemented to obtain the right number of negative literals. To avoid the early convergence of the algorithm, the size of the neuron population is set to 100. The higher number of neurons will lead to a high risk of gaining optimal global solutions [38]. When the ratio of negative literals during the logical phase is determined, the number of negative literals is also confirmed. Tables 1–3 show the list of parameters that are used in each phase of weighted *r*3SAT. The proposed model includes three components: the learning phase, retrieval phase, and testing phase. Table 1 shows the parameters involved in the logic phase. Tables 2 and 3 present the parameters used in the learning phase and the testing phase. ES will be implemented in the logic phase to obtain the correct number of negative literals. The number of η can be obtained by the rounding function of $[rN]$ while the literals mentioned in the retrieval and testing phase are both generated randomly.

Table 1. List of parameters used in W_{r3SAT}.

Parameter	Parameter value
Number of neurons, N	$15 \le N \le 100$
Weighted ratio, r	$[0.1, 0.9]$[36]
Step of the ratio, Δr	0.1 [24]
Number of negative literals, η	$[rN]$
Number of combinations, N_C	100 [37]
Number of learning, N_l	100 [37]
Learning iteration, N_i	$N_i \le N_L$ [24]
Synaptic weight method	Wan Abdullah method (WA) [11]
Number of testing trials N_t	100 [37]
Neuron states in the learning phase	Random 1 or -1
Neuron states in the testing phase	Random 1 or -1
Tolerance value, Tol	0.001 [23]
Active function	HTAF [32]

Table 2. List of symbols used in the learning phase.

Parameter	Parameter value
f_N	Total fitness number
f_C	Current number of fitness
W_i	Synaptic weights obtained currently
W	Synaptic weights obtained by WA method
N_W	Number of synaptic weights each time
N_{TW}	Total number of synaptic weights

Table 3. List of symbols used in the testing phase.

Parameter	Parameter value
$L_W^{\min}$	Minimum energy
L_W	Lyapunov Energy
N_{TS}	Total number of solutions
N_{GS}	Number of global solutions
N_{LS}	Number of local solutions
N_T	Number of testing trials

The ratios of value r have been preset between 0.1 and 0.9, and the step size is set to $\Delta r = 0.2$. The research by Sidik et al. [39] has shown that the selection of the step size plays a crucial role in the process. Additionally, the alignment of the step size with the overall method is essential for ensuring optimal performance. While $r = 0$ means there are no negative literals in the logic structure, $r = 1$ determines all the literals are negative in the logic phase in W_{r3SAT}. The activation function used in the testing phase is the non-linear HTAF because of its discriminability and compatibility with bipolar entries.

5.3 Performance metrics

For each phase of this work, various metrics are utilized to value the performance of the model. The performances of all these models are evaluated by variables of metrics that are well known. As each phase has a different objective, each objective will be illustrated for the actual performance of the proposed model. To obtain the correct synaptic weights of W_{r3SAT}, the cost function must be minimized. The quality of the final states can be evaluated by Eq (28) in the case of the quantity of global minimal solutions. Note that, the smaller the cost function, the better the logic structure. Additionally, the bigger the ratio of global solutions, the better. The optimal value of best fitness in the logic phase is related to the structure of the logic W_{r3SAT}. Mean absolute error (MAE), root means square error (RMSE), and mean absolute percentage error (MAPE) are well known and are widely used to assess the measurement of the performance of logic models [40].

These metrics can measure the effect of the performance of the models and systems. MAE seems a more direct measurement of error. MAE is less affected by outliers and is relatively simple to calculate. As MAE is less sensitive to extreme values, it can provide a more stable evaluation of model performance, especially when the data has uneven distribution or contains noise [41]. MAPE presents the percentage value of the proposed model [42]. RMSE is more sensitive to larger errors. RMSE has several mathematically advantageous properties. For example, it is expressed in the same units as the data, making the scale of the error more intuitively apparent in practical applications. RMSE is one of the standard methods for evaluating many statistical and machine learning models, thus, it has general applicability in model comparison and selection. The lower the value of these errors, the better the learning and retrieval phase. The effectiveness of the logic phase can be investigated in Eqs (29)–(31) as below:

$$MAE_l = \sum_{C=1}^{N_i} \frac{|f_N - f_C|}{N_i}, \tag{29}$$

$$RMSE_l = \sum_{C=1}^{N_i} \sqrt{\frac{(f_N - f_C)^2}{N_i}}, \tag{30}$$

$$MAPE_l = \sum_{C=1}^{N_i} \frac{100}{N_i} \frac{|f_N - f_C|}{|f_N|}, \tag{31}$$

where f_N represents the best fitness of the logic phase and f_C represents the current fitness. N_i denotes the current number of learning trials. Equation (29) evaluates the effectiveness of the average of the performance for literals which can achieve the situation of $f_N = 0$. The discussions regarding the error are RMSE and MAPE, which are shown in Eqs (30) and (31). Both formulations can be used to evaluate the effectiveness of the fitness.

The metric MAE is a standard error based on the average difference between the computed fitness values and the expected fitness of the model we have mentioned. One of our aims is to focus on the effectiveness of W_{r3SAT} logic model by different ratios of negative literals for total. To highlight the variations across different phases of the logic, the performance of each evaluation metric was analyzed. RMSE and MAPE, an extended version of MAE that incorporates percentage values, were discussed across the learning, training, and testing phases.

In the learning phase, the best fitness will be gained when Eq (16) is equal to zero. Additionally, the lower the value of such metrics mentioned above, the better the logic model will be. When the effectiveness of the logic phase is evaluated by Eqs (29)–(31), we can also measure the fitness of neuron states and the clauses that generate the optimal synaptic weights. These metrics are all used to assess the performance of the training phase to evaluate the errors in synaptic weight analysis. Equations (32)–(34) shows the parameters used in the training phase as below:

$$MAE_W = \sum_{C=1}^{N_i} \frac{|W - W_i|}{N_W}, \tag{32}$$

$$RMSE_W = \sqrt{\frac{1}{N_{TW}} \sum_{i=1}^{N_i} (W - W_i)^2}, \tag{33}$$

$$MAPE_W = \sum_{i=1}^{N_i} \frac{100}{W \times N_W} |W - W_i|. \tag{34}$$

W_i is the synaptic weight obtained currently and W is the synaptic weight obtained by the WA method. N_W is the number of synaptic weights obtained during the retrieval phase each time. N_{TW} is the total number of synaptic weights. The energy analysis is used to determine the efficiency of the weighted *r*3SAT in the testing phase, and the RMSE, MAE, and Global ratio are all used to evaluate the errors. Worth mentioning that, the smaller error leads to better logic and also a higher global optimization. When the signal N_{GS} denotes the number of global solutions and N_{LS} denotes the number of local solutions, the logic model is considered satisfied when the error values in Eqs (29) to (34) are equal to zero. In addition, the metrics determined in Eqs (35)–(37) showed the effectiveness of the parameters used in the retrieval phase.

$$MAE_r = \frac{1}{N_T}\sum_{i=1}^{N_T} N_{LS}\,, \tag{35}$$

$$RMSE_r = \sqrt{\frac{1}{N_T \times N_{LS}}\sum_{i=1}^{N_T}\left(N_{LS} - N_{GS}\right)^2}\,, \tag{36}$$

$$GR = \frac{1}{N_T \times N_C}\sum_{i=1}^{N} N_{GS}\,. \tag{37}$$

Here, N_T is the number of testing trails. $N_T \times N_C$ is the total number of solutions. N_{GS} shows the number of solutions that can achieve the global minimum energy in the logic model. To make certain that the final neuron state can be satisfied, the global minimal solutions must be noticed. Guo et al. [25] have proposed a metric named Global Minimum Ratio (GR) to evaluate the energy of the network in recent work. The higher the value of GR and the lower value of the parameters mentioned in Eqs (35) and (36) mean a better satisfiability of the model and also an optimal model.

5.4 *Similarity analysis*

Analyzing the similarity index can evaluate the quality of the final states of the model. The Jaccard Index (JAC) is a ratio of similarity between two distinct data sets. The calculation of the JAC is straightforward and computationally efficient, making it suitable for large-scale datasets. Fletcher et al. [43] used the JAC to measure the similarity between sets of patterns by converting each pattern into a single element within the set. The measurement of the work of [43] focuses on providing conceptual simplicity, computational simplicity, interpretability, and wide applicability. JAC reflects the ratio of the intersection to the union of the two sets, making it intuitive and easy to understand in practical applications [44]. It is used to evaluate the global solutions in the proposed approach. The similarity index JAC is defined in Eq (38) as follows.

$$J\left(S_i^{\max}, S_i\right) = \frac{\alpha}{\alpha + \beta + \gamma}\,. \tag{38}$$

Note that α is the number of states where $\left(S_i^{\max}, S_i\right)$ when both the elements are equal to 1. β is the number of the set when $S_i = -1$ and $S_i^{\max} = 1$. γ is the number of $\left(S_i^{\max}, S_i\right)$ when

$S_i = -1$ and $S_i^{\max} = 1$. Since the similarity index serves as a measure for evaluating the quality of the final states retrieved, the proposed model will utilize this formulation of the similarity index [45] to assess the final neuron states. As previously mentioned, the Jaccard (JAC) similarity analysis will be employed to compare the neuron states, as outlined in Table 4.

Table 4. Variables for similarity index JAC.

Variable	$S_i^{\max}$	S_i
α	1	1
β	1	−1
γ	−1	1

5.5 Baseline logics

To evaluate the performance of the proposed logic, the formulated logic model is compared with the baseline SAT models mentioned below:

(a) Random 2 Satisfiability (RAN2SAT) [22]: This study was first proposed by Sathasivam et al. and was mentioned as the earliest work on the non-systematic logic rule. RAN2SAT consists of both first and second-order logic clauses. This approach broadens the potential to transition from systematic to non-systematic logic, with the latter being more applicable for real-world implementation. Consequently, no work was performed on the measurement of negative literals, nor was there a logic phase involved.

(b) Random 3 Satisfiability (RAN3SAT) [23]: RAN3SAT is an expanding work of RAN2SAT which concluded third-order logic in the model. It consisted of three components: The first, second, and third-order clauses. The work by Karim et al. [23] noticed three types of logic combinations with different k. The literals were generated randomly while the clauses could be defined by the user. This approach significantly enhanced RAN2SAT by incorporating 3SAT, thereby making the structure more flexible. The researchers addressed the limitations of RAN2SAT in existing non-systematic logic, but it did not account for measuring the number of negative literals in the logical model.

(c) G-type Random k Satisfiability (GRAN3SAT) [26]: GRAN3SAT was first put forward by Gao et al. [26] as a flexible logical rule by combining the structure of both systematic and non-systematic logic. It was enumerated randomly with a higher order logical structure including the first-order, second-order, and third-order of clauses. The present research GRAN3SAT can be reduced to systematic logic 2SAT, 3SAT. We focused on the flexibility of the logical structure but overlooked the consideration of the weight of the negative literals.

(d) Weighted Random k Satisfiability (r2SAT) [28]: The work proposed by Zamri et al. considered a class of non-systematic satisfiability logic with the inclusive ratio of negative literals. The approach emphasizes the distribution of negative literals using GA to generate the logic phase. However, this paper only considers the weighting of the structure in first- and second-order clauses. Notably, third-order logic has not been considered in the proposed framework.

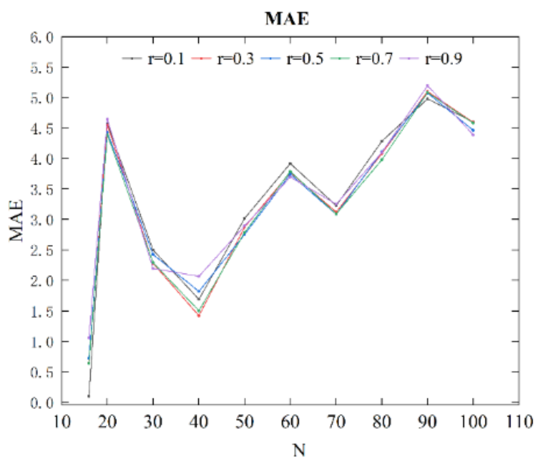
6. Results and discussion

The proposed logic model was compared with state-of-the-art logic models in various aspects during the following work. Each phase of the proposed model will be assessed. Including the learning phase (logic phase), training phase, and the testing phase. ES is embedded into the logic phase to obtain the best value and the correct number of negative literals. It will be used to calculate the right number η with different ratio r, respectfully. The ratio of negative literals mentioned herein results different cases to compare with each other. For example, when $r = 0.3$, it means the negative literals have a percentage of 30% while the other 70% are positive. After the logic phase W_{r3SAT} was generated, W_{r3SAT} was able to be trained to minimize the cost function embedded in DHNN. The metrics MAE, RMSE, and MAPE have been introduced in Eqs (29)–(31) and were used to evaluate the effectiveness of the logic model. To avoid the complexity of the comparison, the number of literals was set up to 100. This drawback of the number controlling is tolerant of a new logic that was first proposed. After the generation of the logic phase, the obtained logic was trained in DHNN. Equations (32)–(34) can measure the effectiveness of the synaptic weight. Equations (35)–(37) was used to evaluate the quality of the proposed model in reaching the global minimum solutions. The comparison between the proposed model and exit SAT models in DHNN was analyzed. Moreover, the similarity of the benchmark station and the final neuron station is measured in terms of the JAC similarity index.

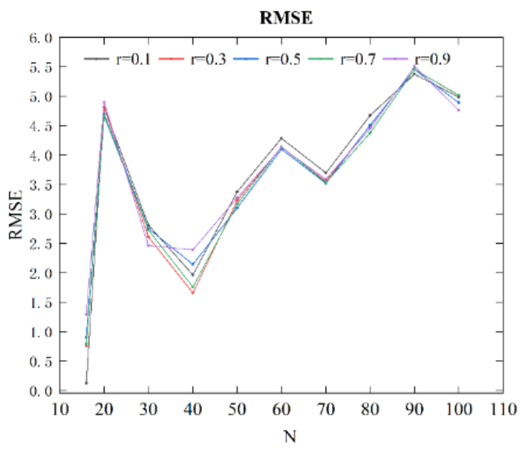
6.1 Analysis of logic phase

The purpose of the logic phase is to find a correct logic structure that has a strong ability for consistent interpretation. The proposed approach utilized ES to gain the number of negative literals for different ratio sets. The learning phase of the proposed model is specifically designed to achieve the desired number of negative literals, as formulated in Eq (15). It has been noted that *r*2SAT is unable to retrieve the correct structure when the number of neurons exceeds 45 for any negative ratio [28]. Due to the utilization of ES, the proposed work can obtain an optimal structure for all r. Herein, the performance metrics MAE_l, $RMSE_l$, and $MAPE_l$ shown in Eqs (29)–(31) are computed to measure the effectiveness of the clause fitness.

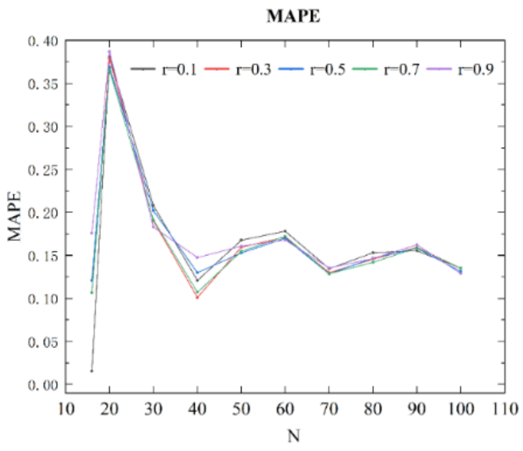
As the performance metrics were used to analyze the fitness of neuron states, the value of these metrics can also be referred to as the fitness error. Note that the main object of the learning phase is to search for the optimal ratio of negative literals that can achieve optimal fitness. Figure 4 illustrates the comparative effectiveness of different ratios in the learning phase in the logic W_{r3SAT}. The proposed value for a consistent step of ratio is determined as the symbol of Δr. To better observe the overall situation, we set the step of ratio equal to 0.2. Figure 4 was generated based on the data calculated under this condition shows that it can be concluded that the ratio $r = 0.7$ plays a good view in the proposed model while the step of ratio $\Delta r = 0.2$. It can achieve a smaller error to get the best fitness. Generally, when the number of neurons is below 20, the errors increase linearly as the number of neurons grows, except for certain values of r. However, when the number of neurons is between 20 and 40, the errors decrease. As the number of neurons continues to increase, the fitness error also increases. This trend has been observed and is consistent with the findings of Karim et al. [23] and Zamri et al. [28].



(a)



(b)



(c)

Figure 4. Metrics analysis based on different ratios within the step $\Delta r = 0.2$.

Figure 4(a) represents the mean absolute error (MAE) during the learning phase, while Figure 4(b) illustrates the root mean squared error (RMSE) and Figure 4(c) demonstrates the mean absolute error (MAPE) during the learning phase with different value of the ratio. Although the differences based on different ratios has shown quite diminish when the number of neurons increased, it can be demonstrated that the ratio of negative literals plays a great part in the contribution of logics. It is worth mentioning that when the ratio gets its extreme value in the range of $[0.1, 0.9]$, the solutions of the combination would not be very multiple. For example, when $r = 0.1$ or $r = 0.9$ and the number of literals is set to 15, the number of negative literals should be counted. The number of negative literals would equal to 1 under the condition of $r = 0.1$ and equal to 13 when the ratio $r = 0.9$. With no ordering arrangement of the literals where the negative ones should be, the possible combinations of W_{r3SAT} is $\binom{15}{1} = \frac{15}{1\,!} = 15$ or $\binom{15}{13} = \frac{15 \times 14 \times 13 \times \cdots \times 3}{13!} = 105$ respectively. When the value $r = 0.5$, the number of negative literals would be calculated and is equal to 7, while the number of possible W_{r3SAT} would increase to $\binom{15}{7} = 6435$. As the number of literals increased, the combinations would increase sharply. These differences in the structure are based on the division of negative literals. A huge amount of partition of literals would cause a huge number of states that must be noticed. In the study of this work, the set of combinations is set up randomly, and the states of the logic are selected randomly also. The evolutionary algorithm ES works effectively when the number of literals is small. More iterations are needed in the learning phase to generate the right logic when the size of neurons is large. It can be noticed that when the work was focused on the specific and exact direction where the negative literals are, there would be a large probability leading to a mismatch in the approximating process. In the work of Zamri et al. [28], once the process is unable to minimize the near-optimal solution, the generated logic fails to proceed to the training phase of DHNN. The proposed model did a good work of solving this confusing by selecting both the logic structure and negative literals randomly. After calculating the number of negative literals, the direction of the negative ones was distributed randomly and as the number of first or third-order clauses. In addition to the analysis of the learning phase, which aims to achieve a correct logic phase that leads to a zero-cost function, it can be concluded that the model is capable of generating the optimal structure of W_{r3SAT} with each value of r mentioned in the present research.

Table 5 shows the list of MAE for the proposed logic model W_{r3SAT} and other models that are used to compete in the learning phase. The following work shows that all the logics mentioned are non-systematic logics such as RAN2SAT, RAN3SAT, GRANSAT, and *r*2SAT. RAN2SAT included the following situations of clauses with first-order and second-order. RAN3SAT set an implementation of third-order clauses into RAN2SAT, which is higher ordered. GRAN3SAT did the work that the logic model has clauses set up with first-order, second-order, or both of these two orders. In the case of added weight which is the ratio of negative literals in the logic rules included first and second-order clauses. W_{r3SAT} has the capability to contain the clauses both first and third-order. It has both positive and negative literals also. Although the ratio of the negative literals was first determined in the logic named *r*2SAT [28], it was the first time adding third-order clauses in the structure. Furthermore, the direction of negative literals was selected randomly. This random selection would make a more flexible model which can be better utilized in real-life implications. The comparison with these non-systematic models shows the drawback of the roles first-order clauses played in the logic rules.

Table 5. Matrix diagram of MAE_l for different logic models. The bolded value signifies the best value of the given metric under the condition of different numbers of neurons.

Number of neurons		16	20	30	40	50	60	70	80	90	100
	r=0.1	**0.0933**	4.5723	2.4977	1.6917	3.0197	3.9192	3.2280	4.2865	4.9791	4.5998
	r=0.2	0.4141	1.6749	**1.9935**	1.8017	**2.7331**	3.7304	3.5173	4.2128	4.9947	4.4834
	r=0.3	0.6426	4.5481	2.2829	**1.4180**	2.8739	3.7704	3.1205	4.0824	5.0996	4.5948
	r=0.4	0.5264	**1.3678**	2.2057	1.6892	2.8247	3.7961	3.2716	4.1188	5.0310	4.5163
*r*3SAT	r=0.5	0.7259	4.4284	2.4265	1.8199	2.7546	3.7430	3.0869	4.1111	5.0755	4.4643
	r=0.6	0.9004	1.4550	2.5154	2.0124	2.7623	3.7089	3.0614	4.1262	5.0890	4.4748
	r=0.7	0.6389	4.3911	2.2934	1.5009	2.7850	3.7880	3.0919	3.9801	5.0839	4.5833
	r=0.8	1.3204	1.3837	2.5303	1.5244	2.9659	**3.6060**	**2.9906**	**3.8111**	**4.9517**	4.4079
	r=0.9	1.0557	4.6448	2.1968	2.0677	2.8949	3.6966	3.2516	4.0996	5.1973	**4.3898**
	RAN2SAT	4.1203	5.5092	5.2476	5.7118	7.7610	8.2361	9.4743	10.7543	12.0106	13.2734
	RAN3SAT	0.8379	4.4673	2.3903	1.7272	2.7969	3.7504	3.1268	4.1517	5.0920	4.5191
	GRAN3SAT	2.1387	3.5202	7.2670	9.4195	13.2555	17.2441	13.7019	15.0012	10.0350	14.3730
	*r*2SAT(r=0.6)	3.2892	4.6438	7.4864	8.7559	17.5114	22.4759	9.5193	17.4561	22.4888	25.2883
	Average	1.2849	3.5851	3.3333	3.1646	5.1491	6.5742	4.9571	6.4763	7.3175	7.5360

The marked values which are in highlighted values represent the best solution of the learning phase which has the lowest errors under the condition of different numbers of neurons. As the performance metric MAE_l can utilized to evaluate the different fitness of the neuron state, the value of MAE_l refers to fitness error. Table 5 demonstrates the value MAE_l of the proposed model W_{r3SAT} and the comparison with other non-systematic logic rules. Especially for the condition of *r*2SAT, the value in the table shows only under the condition of $r = 0.6$. The following tables are the same as above. The value of MAE_l increased when the number of neurons increased. That means it is more difficult to obtain the optimal solutions satisfied $E_{W_{r3SAT}} = 0$ with a high value of *N*. When the number of neurons is smaller than 20, under the condition of $r = 0.1$ can generate the best model of logic which means it can obtain the optimal fitness obviously.

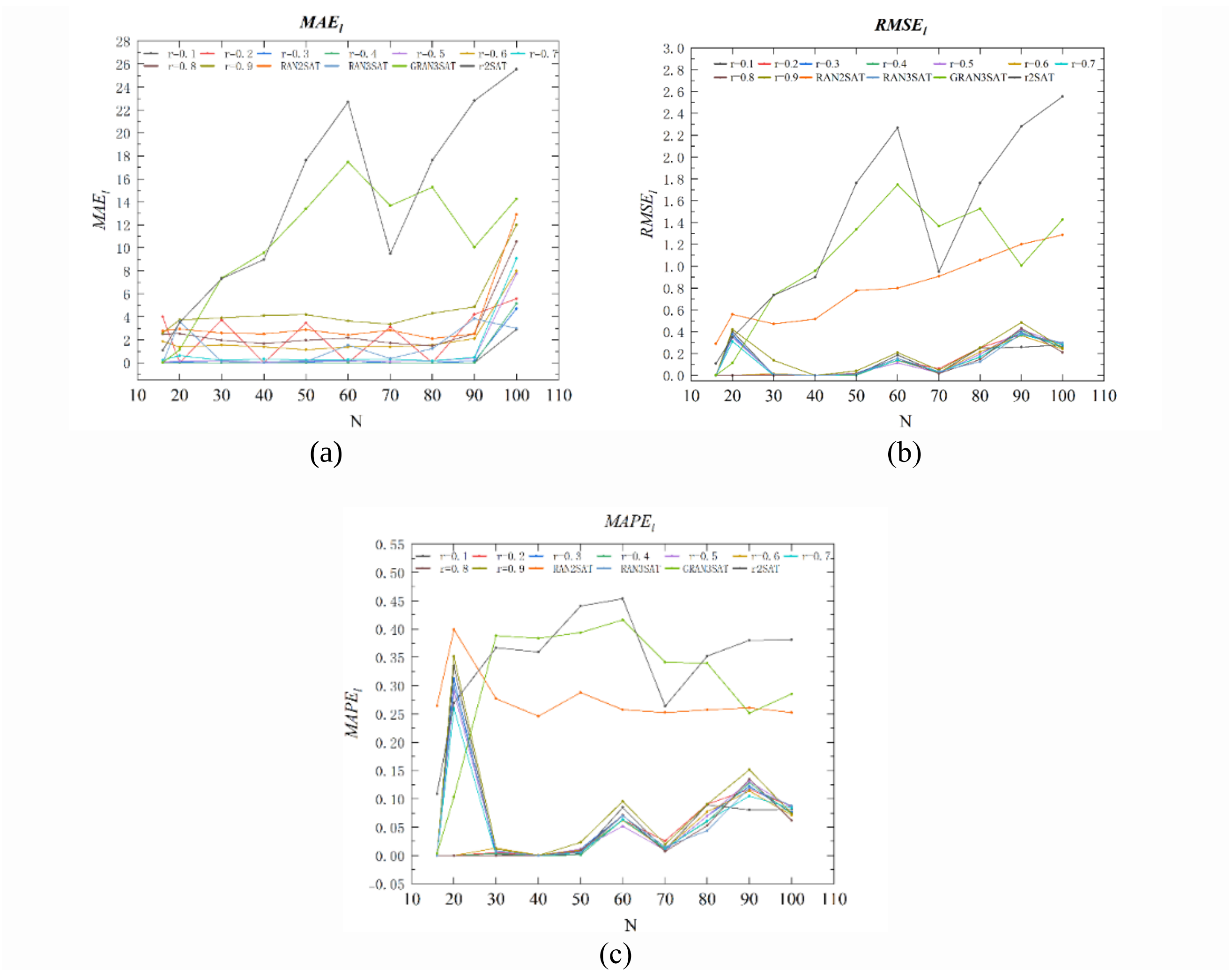


Figure 5. (a) Mean absolute errors (MAE_l), (b) root mean squared error ($RMSE_l$) and (c) mean absolute percentage error ($MAPE_l$) for each ratio in W_{r3SAT} model and some other logic rules.

Figure 5 shows the value of different metrics for each ratio of W_{r3SAT} and the average value of these metrics for *r*2SAT in the cases of all the ratios during the learning phase. While higher MAPE indicates a lower degree of accuracy, the smaller value of MAPE can obtain a better forecast [46]. The errors for each ratio of W_{r3SAT} are relatively close due to the similar structure of first and third-order logic. When the number of neurons is below 30, all the logics have an instability veritable MAPE. When the number of literals increases, the error tends to be more linearly increasing no matter with the structure of the models. Putting this matter of adding ES aside, all the logics can achieve a lower error when the number is bigger than 60 which can be concluded to emphasize a better synaptic weight management. Although *r*2SAT performs well in incorporating the ratio of negative literals during the learning phase, it achieves a MAPE value close to 1. A MAPE value of 1 indicates a significant difference between the maximum fitness and the current fitness [23]. Achieving a smaller MAPE percentage is a key objective during the learning phase for all logic, particularly for the proposed model W_{r3SAT}.

6.2 *Analysis of learning phase*

The analysis of the average errors for each metric indicated that the proposed model did better work than most of the existing models. Figure 6 Displayed the average value of each error metrics. There is only one model named RAN3SAT that is competitive with these W_{r3SAT} models. It is noticeable that the logic RAN3SAT is more consistent than the other non-systematic models shown above except the W_{r3SAT} with ratio $r = 0.2$, 0.4, 0.6 and 0.8 in Figure 5. The phenomenon is interesting that when the ratio of negative literals is set to be $r = 0.3$, 0.5, 0.7 and 0.9 the average value seems the same with RAN3SAT. In the work of [47], it was emphasized that a constrained number of learning iterations does not improve the training phase to achieve higher fitness of neuron states. Therefore, the proposed logic is expected to benefit this study.

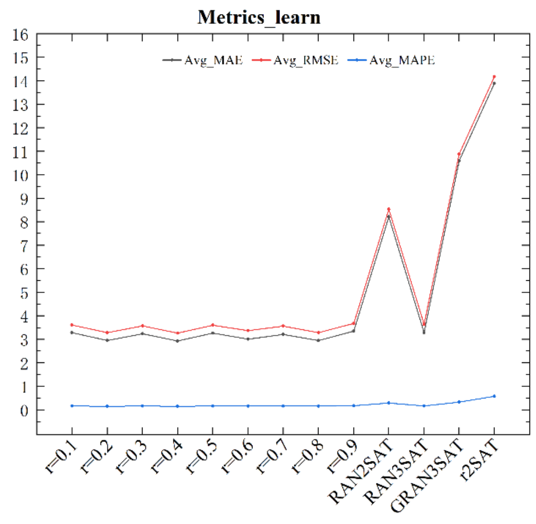
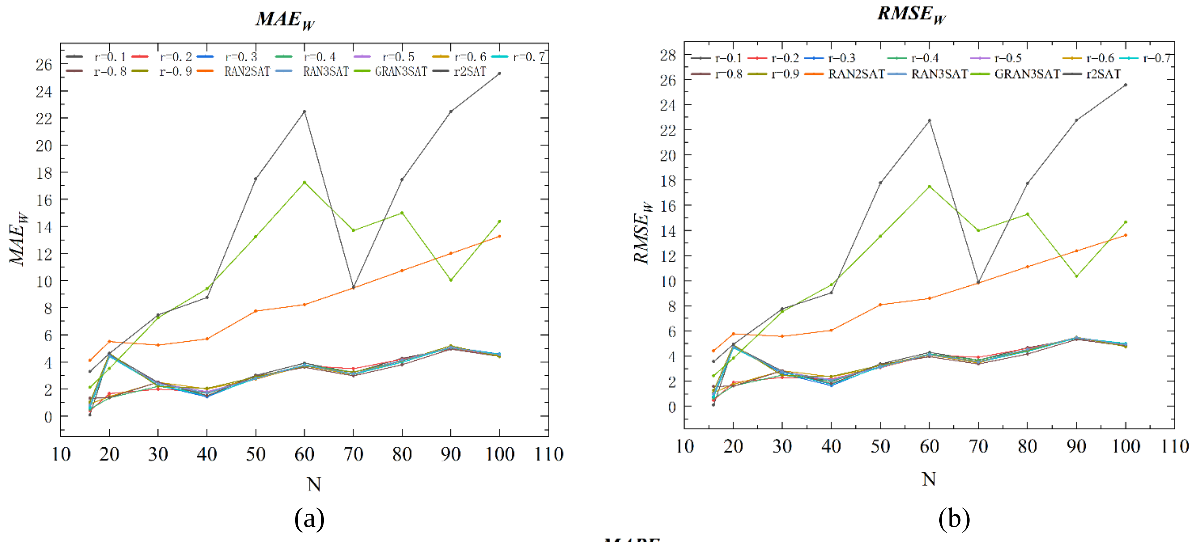


Figure 6. The average value of each error metric for all models in the learning phase.

6.3 Analysis of retrieval phase

Our goal of this section is to discuss and evaluate the effectiveness of the proposed model and different models during the retrieval phase of DHNN. The results of the analysis during the retrieval phase are presented in the next part. Four metrics are presented to evaluate the performance of retrieval errors upon the proposed logic rule and other non-systematic rules. The value of global minimum solutions (GR) can achieve totally near 1 with a low value of *N*. This is because the lower number of clauses tends to be satisfied in the learning phase. As the number of literals increases, RAN2SAT and *r*2SAT are diminished straightly in terms of the literals selecting are random and both of the logic have difficulties to be satisfied. Furthermore, the value of GR quickly reduced to almost zero when the number of neurons increased to more than 50. As mentioned in the work of Guo et al. [25], the value of GR would crash because of the first-order clauses generated randomly. This shows more learning iterations are acquired to satisfy the interpretation of this logic. Figure 7 depicts the values of MAE (a), RMSE (b) and MAPE (c) for all logic models mentioned here. These metrics are utilized to analyze the effectiveness of different ratios of W_{r3SAT} and all the other models. ES was also used in the learning phase to search for fitness. It is worth mentioning that ES was performed on a trial-and-error basis. Based on Figure 7, the conclusion is that when the ratio of W_{r3SAT} is $r = 0.2$, 0.4, 0.6, and 0.8, the errors are almost zero when $N \leq 20$. On the other hand, the errors are not very high when the number of neurons is not bigger than 40. Additionally, the fluctuation waved strongly when *N* increased. As the number of neurons increases, the capability of DHNN leads to the global optimal solutions decrease. This causes DHNN to store the synaptic weight randomly which will lead the final states to local minimum solutions. As the results are shown in Figure 7, the errors are almost nearly zero when $N \leq 40$ for all the corresponding W_{r3SAT} models. Besides, RAN3SAT obtained the right synaptic weights take contrast with RAN2SAT and *r*2SAT. GRAN3SAT also expressed a good performance in that situation. There is an interesting observation that all the values of GR are within a high level when the number of neurons is no more than 50 except RAN2SAT and *r*2SAT. For the proposed model W_{r3SAT}, the value is at a high level even though the number has increased to 70. These can be seen in Table 6. Where Table 7 shows the values of MAPE for different logic structures. Overall, the graph in Figure 8 and the values shown in Table 8 demonstrate that after adding the ratio of negative literals better GR can be achieved compared to other logics. It is noteworthy that the values in the table for *r*2SAT are only expressed when the ratio is $r = 0.6$ while the figure shows the average value of all the ratio contributions.

In the simulations for the proposed model, both the number of possible combinations and the distribution of negative literals were randomly selected. As the number of neuron states increases, achieving a zero-cost function that yields the optimal solutions becomes increasingly challenging. The proposed model can be improved by a high number of learning and then achieve the maximum fitness of neuron states.



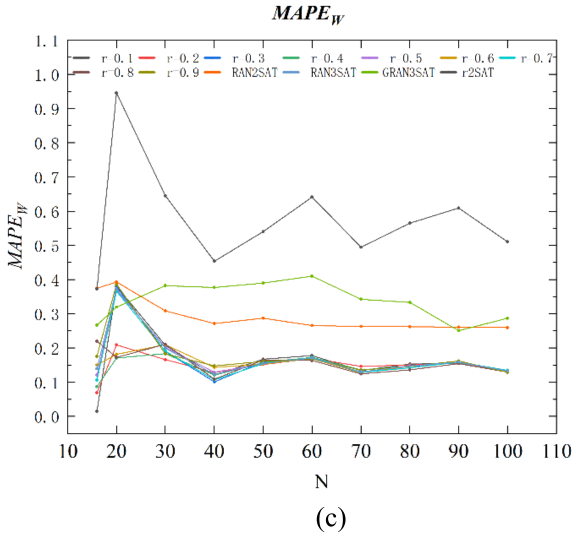


Figure 7. MAE, RMSE and MAPE for all logic rules in the training phase.

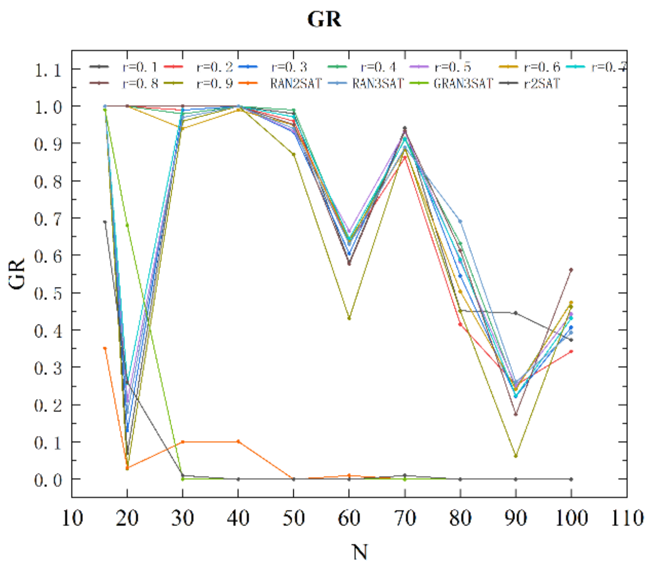


Figure 8. The values of GR for all DHNN models.

Table 6. Matrix diagram of $RMSE_W$ for different logic models. The bolded value signifies the best value of the given metric under the condition with different numbers of neurons.

Number of neurons		16	20	30	40	50	60	70	80	90	100
	r=0.1	**0.1219**	4.8144	2.8113	1.9643	3.3750	4.2869	3.6962	4.6755	5.3745	4.9855
	r=0.2	0.4982	1.9208	**2.3093**	2.1896	**3.0808**	4.0911	3.9165	4.6163	5.4115	4.8769
	r=0.3	0.7587	4.8137	2.6073	**1.6543**	3.2270	4.1317	3.5522	4.5034	5.4579	5.0133
	r=0.4	0.6526	1.6289	2.4953	1.9975	3.1982	4.1688	3.6899	4.5113	5.4257	4.9413
*r*3SAT	r=0.5	0.9039	4.6970	2.7521	2.1419	3.0995	4.1016	3.5170	4.5114	5.4539	4.8904
	r=0.6	1.1225	1.7550	2.8433	2.3590	3.1474	4.0805	3.4596	4.5292	5.4971	4.8980
	r=0.7	0.7929	4.6605	2.7280	1.7547	3.1647	4.1422	3.5312	4.3769	5.4555	5.0165
	r=0.8	1.5820	**1.6241**	2.8520	1.8163	3.3376	**3.9526**	**3.3824**	**4.1792**	**5.3260**	4.8067
	r=0.9	1.2880	4.8953	2.4618	2.3898	3.2742	4.1276	3.5910	4.4522	5.5034	**4.7648**
	RAN2SAT	4.4254	5.7775	5.5660	6.0513	8.0910	8.5891	9.8198	11.1049	12.3714	13.6176
	RAN3SAT	1.0351	4.7431	2.7464	2.0675	3.1917	4.1189	3.5380	4.5437	5.4831	4.9206
	GRAN3SAT	2.4459	3.8424	7.5350	9.6866	13.5325	17.5066	13.9830	15.2899	10.3519	14.6764
	*r*2SAT(r=0.6)	3.5785	4.9457	7.7696	9.0438	17.7719	22.7384	9.8698	17.7500	22.7728	25.5747
	Average	1.4774	3.8553	3.6521	3.4705	5.4993	6.9258	5.3497	6.8495	7.6834	7.9217

Table 7. Matrix diagram of $MAPE_W$ for different logic models. The bolded value signifies the best value of the given metric under the condition with different number of neurons.

Number of neurons		16	20	30	40	50	60	70	80	90	100
	r=0.1	**0.0156**	0.3810	0.2081	0.1208	0.1678	0.1781	0.1345	0.1531	0.1556	0.1353
	r=0.2	0.0690	0.2094	**0.1661**	0.1287	**0.1518**	0.1696	0.1466	0.1505	0.1561	0.1319
	r=0.3	0.1071	0.3790	0.1902	**0.1013**	0.1597	0.1714	0.1300	0.1458	0.1594	0.1351
	r=0.4	0.0877	**0.1710**	0.1838	0.1207	0.1569	0.1726	0.1363	0.1471	0.1572	0.1328
*r*3SAT	r=0.5	0.1210	0.3690	0.2022	0.1300	0.1530	0.1701	0.1286	0.1468	0.1586	0.1313
	r=0.6	0.1501	0.1819	0.2096	0.1437	0.1535	0.1686	0.1276	0.1474	0.1590	0.1316
	r=0.7	0.1065	0.3659	0.1911	0.1072	0.1547	0.1722	0.1288	0.1421	0.1589	0.1348
	r=0.8	0.2201	0.1730	0.2109	0.1089	0.1648	**0.1639**	**0.1246**	**0.1361**	**0.1547**	0.1296
	r=0.9	0.1759	0.3871	0.1831	0.1477	0.1608	0.1680	0.1355	0.1464	0.1624	**0.1291**
	RAN2SAT	0.3746	0.3935	0.3087	0.2720	0.2874	0.2657	0.2632	0.2623	0.2611	0.2603
	RAN3SAT	0.1396	0.3723	0.1992	0.1234	0.1554	0.1705	0.1303	0.1483	0.1591	0.1329
	GRAN3SAT	0.2673	0.3200	0.3825	0.3768	0.3899	0.4106	0.3425	0.3334	0.2509	0.2875
	*r*2SAT(r=0.6)	0.3728	0.9460	0.6453	0.4536	0.5406	0.6417	0.4950	0.5653	0.6094	0.5107
	Average	0.1698	0.3576	0.2524	0.1796	0.2151	0.2325	0.1864	0.2019	0.2079	0.1833

Table 8. The matrix diagram of global ratio (GR). The bolded value signifies the top three in average for all the situations.

Number of neurons		16	20	30	40	50	60	70	80	90	100	Avg
*r*3SAT	r=0.1	1.0000	0.0688	0.9900	1.0000	0.9803	0.5812	0.9408	0.4517	0.4450	0.3733	0.6831
	r=0.2	1.0000	1.0000	0.9900	1.0000	0.9602	0.6317	0.8626	0.4152	0.2525	0.3424	0.7455
	r=0.3	1.0000	0.1300	0.9900	1.0000	0.9302	0.6051	0.9334	0.5453	0.2227	0.4075	0.6764
	r=0.4	1.0000	1.0000	0.9800	1.0000	0.9900	0.6325	0.9131	0.6319	0.2429	0.4743	**0.7865**
	r=0.5	1.0000	0.2100	0.9703	1.0000	0.9334	0.6650	0.9311	0.5832	0.2505	0.4422	0.6986
	r=0.6	1.0000	1.0000	0.9400	0.9900	0.9500	0.6406	0.8847	0.5027	0.2410	0.4736	**0.7623**
	r=0.7	1.0000	0.2572	1.0000	1.0000	0.9717	0.6458	0.9109	0.5884	0.2240	0.4315	0.7030
	r=0.8	1.0000	1.0000	1.0000	1.0000	0.9507	0.5773	0.9341	0.6130	0.1736	0.5612	**0.7810**
	r=0.9	1.0000	0.0277	0.9600	1.0000	0.8707	0.4310	0.8903	0.4500	0.0617	0.4622	0.6154
	RAN2SAT	0.3510	0.0300	0.1000	0.1007	0.0000	0.0100	0.0000	0.0000	0.0000	0.0000	0.0592
	RAN3SAT	1.0000	0.1791	0.9700	1.0000	0.9400	0.6300	0.8904	0.6909	0.2606	0.3932	0.6954
	GRAN3SAT	0.9900	0.6800	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1670
	*r*2SAT(r=0.6)	0.6900	0.2600	0.0100	0.0000	0.0000	0.0000	0.0100	0.0000	0.0000	0.0000	0.0970
	Average	0.9255	0.4494	0.7616	0.7762	0.7290	0.4654	0.7001	0.4209	0.1827	0.3355	0.5746

6.4 Analysis of testing phase

The following section demonstrates how the proposed model will be evaluated based on the management of synaptic weights. The efficiency of the testing phase will be indicated whether W_{r3SAT} can successfully attain the optimal synaptic weights and produce global minimum solutions with the final neuron states. As shown in Table 6, note that the proposed logic model added a negative ratio first, and the value η can be calculated using Eq (14), the proposed model demonstrates the same training method. W_{r3SAT} can attain minimum errors of nearly zero for the ratio $r = 0.2$, 0.4, 0.6 and 0.8 when the number of neurons is less than 50. The same circumstances have happened in the learning phase. Table 7 shows that the proposed logic has the lowest error value of MAPE when the number of neurons increases not less than 60. The error fluctuation occurred when the number of literals is continually increased but not because of the differences in the ratio. The results shown in Figure 9 indicate that the MAE errors for RAN2SAT and *r*2SAT are at a very high level. This is because both logic structures have first and second-order clauses randomly. When the number of neurons increases, especially more than 50, the error can reach 1, so it appears that the global minimum solutions in the final states cannot be obtained. Strictly speaking, the model can obtain global minimum energy based on the optimal synaptic weight. Based on the values shown in Figure 10, RAN2SAT has the worst average value of GR. The reasons for this depend on the low capability of obtaining a consistent interpretation of non-systematic models. Research by Huang et al. [48] suggested that the implementation of improved differential evolution (IDE) can do some advantages in convergence speed and reduce optimization time for global searching in the testing phase. No matter how many neuron literals there are, MAE for testing seems similar between RAN3SAT and W_{r3SAT}. This interesting phenomenon occurs due to the structure of both RAN3SAT and W_{r3SAT} are selected randomly and they both have the fortune to generate high potential of third-order clauses. Moreover, W_{r3SAT} can sometimes be generated as an extremely special item of RAN3SAT. Besides, first-order logic which is one of the important structures in the proposed logic plays a great role in disturbing the process of retrieving and leading to a high level of testing error.

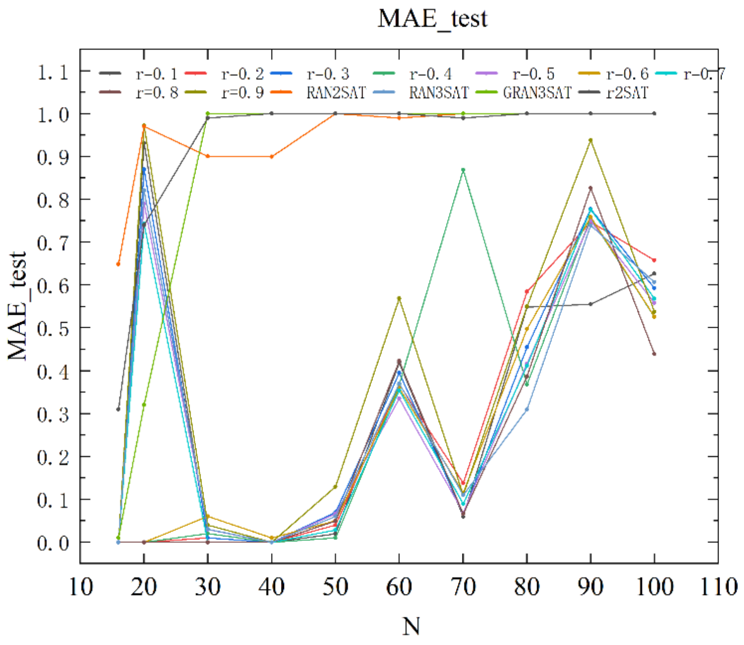


Figure 9. The values of MAE in the testing phase.

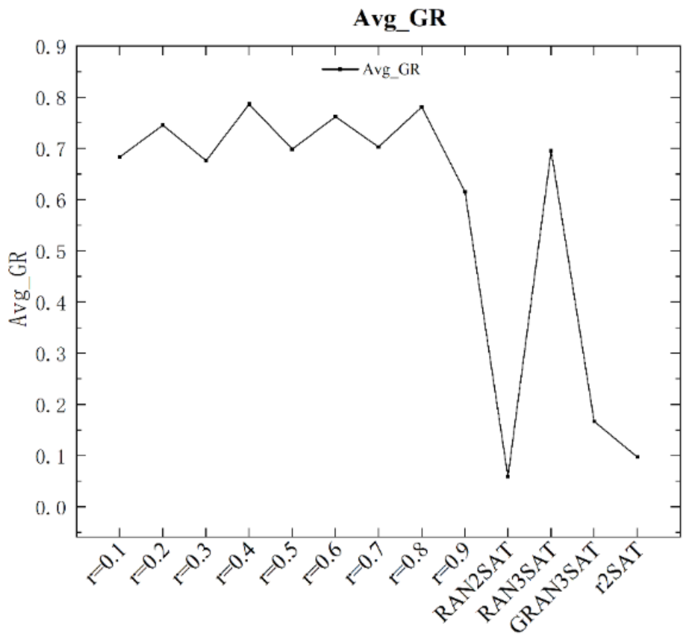


Figure 10. The average values of GR.

There is another interesting point that has been underlined by Karim et al. [23], which is that RAN3SAT with first and third-order clauses has reached the highest error value compared with other logics. However, it is quite different after adding the ratio of negative literals to the proposed logic model. As shown in Figure 9, although W_{r3SAT} has components with first and third-order logic, it does have a lower value of error when the number of neurons grows larger than 30.

6.5 Similarity analysis

The Jaccard similarity index (JAC) can be utilized to check the similarity between different data sets. It has been used to evaluate the global solutions in the work by Bag, S. et al. [49]. It was also suggested to assess the performance of logic with DHNN by the work [50]. JAC is a similarity ratio that measures the quality of final states, making it a standard parameter for indexing. In Figure 11, it can be observed that *r*2SAT has the highest value when $25 < N < 35$ and $65 < N < 75$. Sharp decline happened when the number of neurons increased by more than 40. After the number reached 60, large fluctuations occurred for the logic *r*2SAT. This fluctuation does not only bechance in *r*2SAT but also took place in RAN2SAT and GRAN3SAT for different ranges of neurons. Compared with the existing model, ratio-adding logic W_{r3SAT} has a gentler variation with each mentioned ratio in the proposed model. RAN3SAT has a similar wave compared with the proposed model. The waves of the JAC values for each ratio of the Weighted Random Satisfiability show parallel lines in Figure 11. The oscillations for RAN2SAT, GRAN3SAT, and *r*2SAT are zero at $N = 70$. Even though W_{r3SAT} is satisfied, the proposed model would fail to achieve the optimal states. As the local solutions increase in terms of the neuron increase, the ratio plays an important role in distributing the effectiveness. The simulation stops at $N = 70$ is enough for other baseline logic, it is not abundant for the logic W_{r3SAT}. Thus, the number is settled to 100. It is noteworthy that as the ratio of negative literals increases, the JAC similarity value changes in parallel. The higher the value of the ratio, the larger the similarity index.

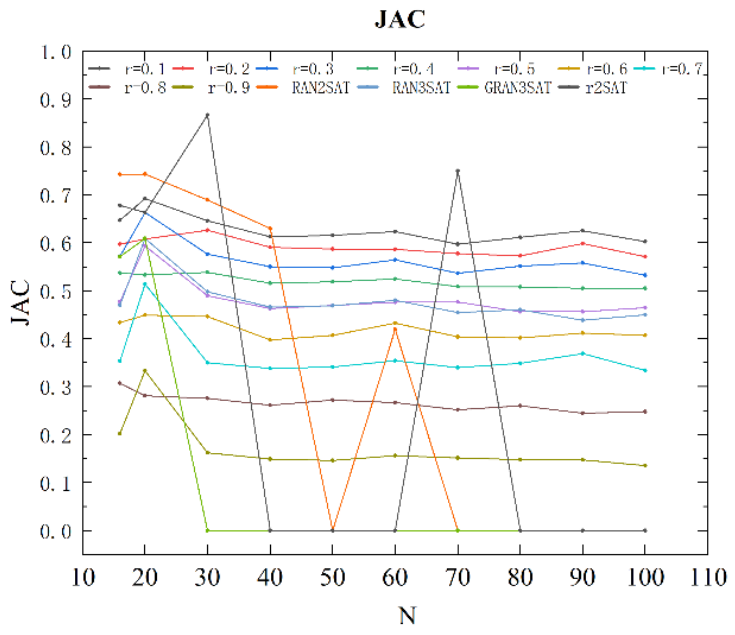


Figure 11. JAC value of each model.

Figure 11 presented the total trend is in the range of $[0.50, 0.65]$ when the ratio $r \in [0.1, 0.4]$. Under the situation $r \in [0.5, 0.8]$, the result reduces to $[0.27, 0.47]$. Furthermore, the value is declined by almost 0.1 when the ratio is up to 0.9. Despite the influence based on the diversification of the ratio in W_{r3SAT}, oscillations happened when the number of neurons is below 30. It needs to be noticed that the frequent oscillations occur as the neurons increase under the condition $r = 0.7$. Moreover, $r = 0.5$ indicates the gentlest diversification. As the metric JAC is aimed at neuron variation produced by W_{r3SAT}, additional similarity index should be invited. The logic rule that achieves the highest global optimum is also the most effective for logic mining. The final neuron state can be converted into the maximum induced logic [23]. The benefits of the proposed logic offer significant advantages for real-life applications.

From the discussion mentioned above, the definition of minimizing the cost function, generating the optimal synaptic weigh, and the variation of final neuron states of W_{r3SAT} were all determined. Adding the ratio of negative literals presents better in obtaining higher global minimum solutions and affirming final neuron states. We outline that when adding the ratio $r = 0.7$ present the best logic model with producing lower errors and higher global minimum solutions. On the other hand, due to the learning phase of algorithm ES, this study limits it to 100 neurons. Overall, we can improve this limitation of the proposed model by adding other metaheuristics such as GA and EA (Election Algorithm). Despite the flexibility of W_{r3SAT} in DHNN, additional modifications can be done to improve the effectiveness of solutions. This means to not only increase the number of iterations but also decrease the errors and more global solutions.

7. Open problems

In the field of artificial intelligence, SAT has long been a foundation of theoretical research. Different logic structures and algorithms for SAT have been proposed and extensively discussed [18,20,35]. However, while higher-order logic has been broadly explored in computer science [20,23,35], higher-order logic adding a negative ratio remains relatively under-explored. Specifically, how to effectively express this negative ratio during the training phase is an unsolved issue. One key problem is how to design new activation functions that can accommodate the property of higher-order logic within DHNN.

Higher-order logic, which contains higher-order clauses, often involves more complex computing power. How to balance the effectiveness of benefit and computation cost remains a difficult task. Current neural network architectures, such as CNN [51], RNN [52], have limitations when the logical structure becomes more complex [53], especially for higher-order logic in high-dimensional spaces. In DHNN which has been discussed herein, the same problem must be solved and the computing time consumption also needs to be considered. Higher-order logic adding a negative ratio would make the network gradients unstable, which could lead to overfitting or gradient vanishing. How to design a model that can keep the properties of the logic in higher order while maintaining stability is a great challenge for further work.

The treatment of Weighted Random Satisfiability with higher-order logic in DHNN is an open problem that leaves a lot of strategies unexplored. The solving of the problems would push the applications of AI in complex reasoning and decision-making systems.

8. Conclusions

We explore a novel logical model named W_{r3SAT}, which was proposed with a different logical structure adding a ratio of negative literals as a new instruction in DHNN. The proposed logic rules provide different ratios and the generated negative literals distributed in each clause are randomly. A logic phase will be introduced to aid the generation of the correct Weighted Random k SAT according to the initiated r before being trained in DHNN. Random literals of Weighted Random k SAT will be generated according to the value of the input ratio r. The value of η will be evaluated according to the value of weight in each clause. Although there are many algorithms for finding the number of negative literals, the minimization will be executed using ES in the logic phase. As the outperformance of the logic has shown that the differences are not quite clear, the step of the ratio chooses the length of $\Delta r = 0.1$. It is worth mentioning that further work can focus on a smaller step and make a more detailed work on the effectiveness of ratio. The results in the training and testing phase portrayed the capability of W_{r3SAT} in producing optimal solutions for all values of r. The optimal model obtains a lower cost function, less error, and higher satisfaction in each step of the model. On the other hand, more diversity was obtained, which is more effective in real-life research. The comparisons between W_{r3SAT} and other non-systematic logic models (RAN2SAT, RAN3SAT, GRAN3SAT, and *r*2SAT) were quite distinct. The results showed that the proposed model W_{r3SAT} can obtain a better performance which indicates that adding a ratio will take advantage of global minimum generations.

For future work, logic mining should be discussed based on the distribution of negative literals which can be set up to the prescribed order of clauses. Researchers have explored log-linear analysis combined with higher-order logic mining during the pre-processing phase [54,55]. Additionally, reverse-based analysis logic mining has been investigated, leading to an increase in storage capacity [56,57]. Furthermore, logic mining and optimization of the learning phase provided improvement for the next step in research, and many other metaheuristic algorithms could be added in the learning phase to improve obtaining the optimal synaptic weight. Additionally, this can generate global minimum solutions with various final neuron states. Moreover, this can be utilized in many real-life problems.

Author contributions

Conceptualization, methodology, software, data curation and writing-original draft, X.L.; resources and formal analysis, Y.G.; validation and investigation, N.E.Z.; supervision and funding acquisition, M.S.M.K.; writing-review and editing, S.A.; visualization, Y.C.; project administration, M.S.M.K. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The authors express special thanks to all researchers in the Artificial Intelligence Research Development Group (AIRDG) for their supports and advises sincerely. This research is funded and supported by Universiti Sains Malaysia, Short Term Grant, 304/PMATHS/6315655.

Conflict of interest

The authors declare no conflicts of interest.

References

1. M. Soori, B. Arezoo, R. Dastres, Artificial intelligence, machine learning and deep learning in advanced robotics, a review, *Cogn. Robot.*, **3** (2023), 54–70. https://doi.org/10.1016/j.cogr.2023.04.001
2. L. Feng, J. Zhang, Application of artificial neural networks in tendency forecasting of economic growth, *Econ. Model.*, **40** (2014), 76–80. https://doi.org/10.1016/j.econmod.2014.03.024
3. A. Nikitas, K. Michalakopoulou, E. T. Njoya, D. Karampatzakis, Artificial intelligence, transport and the smart city: definitions and dimensions of a new mobility era, *Sustainability*, **12** (2020), 2789. https://doi.org/10.3390/su12072789
4. M. Özbey, M. Kayri, Investigation of factors affecting transactional distance in E-learning environment with artificial neural networks, *Educ. Inf. Technol.*, **28** (2023), 4399–4427. https://doi.org/10.1007/s10639-022-11346-4
5. M. Tkáč, R. Verner, Artificial neural networks in business: two decades of research, *Appl. Soft Comput.*, **38** (2016), 788–804. https://doi.org/10.1016/j.asoc.2015.09.040
6. H. Chereda, A. Bleckmann, K. Menck, J. Perera-Bel, P. Stegmaier, F. Auer, et al., Explaining decisions of graph convolutional neural networks: patient-specific molecular subnetworks responsible for metastasis prediction in breast cancer, *Genome Med.*, **13** (2021), 1–16. https://doi.org/10.1186/s13073-021-00845-7
7. J. J. Hopfield, D. W. Tank, "Neural" computation of decisions in optimization problems, *Biol. Cybern.*, **52** (1985), 141–152. http://doi.org/10.1007/BF00339943
8. K. Pagiamtzis, A. Sheikholeslami, Content-addressable memory (CAM) circuits and architectures: A tutorial and survey, *IEEE J. Solid-State Circuits*, **41** (2006), 712–727. http://doi.org/10.1109/JSSC.2005.864128
9. M. Irfan, A. I. Sanka, Z. Ullah, R. C. Cheung, Reconfigurable content-addressable memory (CAM) on FPGAs: a tutorial and survey, *Future Gener. Comput. Syst.*, **128** (2022), 451–465. https://doi.org/10.1016/j.future.2021.09.037
10. A. Alway, N. E. Zamri, M. S. Mohd Kasihmuddin, A. Mansor, S. Sathasivam, Palm oil trend analysis via logic mining with Discrete Hopfield Neural Network, *Pertanika J. Sci. Technol.*, **28** (2020), 967–981.
11. W. A. T. W. Abdullah, Logic programming on a neural network, *Int. J. Intell. Syst.*, **7** (1992), 513–519. https://doi.org/10.1002/int.4550070604
12. Y. Xie, A. Srivastava, Anti-SAT: mitigating SAT attack on logic locking, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, **38** (2018), 199–207. https://doi.org/10.1109/TCAD.2018.2801220
13. F. Ivančić, Z. Yang, M. K. Ganai, A. Gupta, P. Ashar, Efficient SAT-based bounded model checking for software verification, *Theor. Comput. Sci.*, **404** (2008), 256–274. https://doi.org/10.1016/j.tcs.2008.03.013
14. C. Occhipinti, A. Carnevale, L. Briguglio, A. Iannone, P. Bisconti, SAT: a methodology to assess the social acceptance of innovative AI-based technologies, *J. Inf. Commun. Ethics Soc.*, **21** (2023), 94–111. https://doi.org/10.1108/JICES-09-2021-0095
15. C. Castellini, E. Giunchiglia, A. Tacchella, SAT-based planning in complex domains: concurrency, constraints and nondeterminism, *Artif. Intell.*, **147** (2003), 85–117. https://doi.org/10.1016/S0004-3702(02)00375-2

16. H. Youness, M. Osama, A. Hussein, M. Moness, A. M. Hassan, An effective SAT solver utilizing ACO based on heterogeneous systems, *IEEE Access*, **8** (2020), 102920–102934. https://doi.org/10.1109/ACCESS.2020.2999382
17. S. Sathasivam, Upgrading logic programming in Hopfield network, *Sains Malaysiana*, **39** (2010), 115–118.
18. M. S. Mohd Kasihmuddin, M. A. Mansor, M. F. Md Basir, S. Sathasivam, Discrete mutation Hopfield Neural Network in propositional satisfiability, *Mathematics*, **7** (2019), 1133. https://doi.org/10.3390/math7111133
19. J. Zhu, A. Salhotra, C. R. Meinecke, P. Surendiran, R. Lyttleton, D. Reuter, et al., Solving the 3-satisfiability problem using network-based biocomputation, *Adv. Intell. Syst.*, **4** (2022), 2200202. https://doi.org/10.1002/aisy.202200202
20. M. A. Mansor, M. S. M. Kasihmuddin, S. Sathasivam, Artificial immune system paradigm in the Hopfield network for 3-satisfiability problem, *Pertanika J. Sci. Technol.*, **25** (2017), 1173–1188.
21. M. Mouhoub, Systematic versus non-systematic techniques for solving temporal constraints in a dynamic environment, *AI Commun.*, **17** (2004), 201–211.
22. S. Sathasivam, M. A. Mansor, A. I. M. Ismail, S. Z. M. Jamaludin, M. S. M. Kasihmuddin, M. Mamat, Novel random k satisfiability for k≤2 in Hopfield Neural Network, *Sains Malays.*, **49** (2020), 2847–2857. http://doi.org/10.17576/jsm-2020-4911-23
23. S. A. Karim, N. E. Zamri, A. Alway, M. S. M. Kasihmuddin, A. I. M. Ismail, M. A. Mansor, et al., Random satisfiability: a higher-order logical approach in Discrete Hopfield Neural Network, *IEEE Access*, **9** (2021), 50831–50845. http://doi.org/10.1109/ACCESS.2021.3068998
24. A. Alway, N. E. Zamri, S. A. Karim, M. A. Mansor, M. S. Mohd Kasihmuddin, M. Mohammed Bazuhair, Major 2 satisfiability logic in Discrete Hopfield Neural Network, *Int. J. Comput. Math.*, **99** (2022), 924–948. http://doi.org/10.1080/00207160.2021.1939870
25. Y. Guo, M. S. M. Kasihmuddin, Y. Gao, M. A. Mansor, H. A. Wahab, N. E. Zamri, et al., YRAN2SAT: a novel flexible random satisfiability logical rule in Discrete Hopfield Neural Network, *Adv. Eng. Softw.*, **171** (2022), 103169. https://doi.org/10.1016/j.advengsoft.2022.103169
26. Y. Gao, Y. Guo, N. A. Romli, M. S. M. Kasihmuddin, W. Chen, M. A. Mansor, et al., GRAN3SAT: creating flexible higher-order logic satisfiability in the Discrete Hopfield Neural Network, *Mathematics*, **10** (2022), 1899. https://doi.org/10.3390/math10111899
27. A. Darmann, J. Döcker, B. Dorn, The monotone satisfiability problem with bounded variable appearances, *Int. J. Found. Comput. Sci.*, **29** (2018), 979–993. https://doi.org/10.1142/S0129054118500168
28. N. E. Zamri, S. A. Azhar, M. A. Mansor, A. Alway, M. S. M. Kasihmuddin, Weighted Random k Satisfiability for k=1, 2 (r2SAT) in Discrete Hopfield Neural Network, *Appl. Soft Comput.*, **126** (2022), 109312. https://doi.org/10.1016/j.asoc.2022.109312
29. S. Mertens, Exhaustive search for low-autocorrelation binary sequences, *J. Phys. A: Math. Gen.*, **29** (1996), L473. https://doi.org/10.1088/0305-4470/29/18/005
30. C. A. Coello, An updated survey of GA-based multiobjective optimization techniques, *ACM Comput. Surv. (CSUR)*, **32** (2000), 109–143. https://doi.org/10.1145/358923.358929
31. A. H. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, V. B. S. Prasath, Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach, *Information*, **10** (2019), 390. https://doi.org/10.3390/info10120390

32. Z. Wang, Z. Liang, R. Zeng, H. Yuan, R. S. Srinivasan, Identifying the optimal heterogeneous ensemble learning model for building energy prediction using the exhaustive search method, *Energy Buildings*, **281** (2023), 112763. https://doi.org/10.1016/j.enbuild.2022.112763
33. X. Wang, X. Zhang, L. Dong, H. Liu, Q. Wu, R. Mohan, Development of methods for beam angle optimization for IMRT using an accelerated exhaustive search strategy, *Int. J. Radiat. Oncol. Biol. Phys.*, **60** (2004), 1325–1337. https://doi.org/10.1016/j.ijrobp.2004.06.007
34. A. Darmann, J. Döcker, On simplified NP-complete variants of Monotone 3-Sat, *Discrete Appl. Math.*, **292** (2021), 45–58. http://doi.org/10.1016/j.dam.2020.12.010
35. M. A. Mansor, S. Sathasivam, Accelerating activation function for 3-satisfiability logic programming, *Int. J. Intell. Syst. Appl.*, **8** (2016), 44–50. http://doi.org/10.5815/ijisa.2016.10.05
36. R. Ma, Y. Xie, S. Zhang, W. Liu, Convergence of discrete delayed Hopfield neural networks, *Comput. Math. Appl.*, **57** (2009), 1869–1876. https://doi.org/10.1016/j.camwa.2008.10.006
37. O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, H. Arshad, State-of-the-art in artificial neural network applications: a survey, *Heliyon*, **4** (2018), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938
38. A. Nabi, J. Moehlis, Single input optimal control for globally coupled neuron networks, *J. Neural Eng.*, **8** (2011), 065008. https://doi.org/10.1088/1741-2560/8/6/065008
39. S. S. Muhammad Sidik, N. E. Zamri, M. S. Mohd Kasihmuddin, H. A. Wahab, Y. Guo, M. A. Mansor, Non-systematic weighted satisfiability in Discrete Hopfield Neural Network using binary artificial bee colony optimization, *Mathematics*, **10** (2022), 1129. http://doi.org/10.3390/math10071129
40. A. Jierula, S. Wang, T. M. Oh, P. Wang, Study on accuracy metrics for evaluating the predictions of damage locations in deep piles using artificial neural networks with acoustic emission data, *Appl. Sci.*, **11** (2021), 2314. https://doi.org/10.3390/app11052314
41. C. J. Willmott, K. Matsuura, Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, *Climate Res.*, **30** (2005), 79–82.
42. D. Chicco, M. J. Warrens, G. Jurman, The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation, *PeerJ Comput. Sci.*, **7** (2021), e623. https://doi.org/10.7717/peerj-cs.623
43. S. Fletcher, M. Z. Islam, Comparing sets of patterns with the Jaccard index, *Australas. J. Inf. Syst.*, **22** (2018). https://doi.org/10.3127/ajis.v22i0.1538
44. M. Tantardini, F. Ieva, L. Tajoli, C. Piccardi, Comparing methods for comparing networks, *Sci. Rep.*, **9** (2019), 17557. https://doi.org/10.1038/s41598-019-53708-y
45. A. Strehl, J. Ghosh, R. Mooney, Impact of similarity measures on web-page clustering, *AAAI Workshop Papers 2000*, 2000, 58–64.
46. H. K. Sharma, K. Kumari, S. Kar, A rough set approach for forecasting models, *Decis. Making: Appl. Manag. Eng.*, **3** (2020), 1–21. https://doi.org/10.31181/dmame2003001s
47. S. Sathasivam, M. Mamat, M. S. M. Kasihmuddin, M. A. Mansor, Metaheuristics approach for maximum k satisfiability in restricted neural symbolic integration, *Pertanika J. Sci. Technol.*, **28** (2020), 545–564.
48. W. Huang, Y. Li, Y. Huang, Deep hybrid neural network and improved differential neuroevolution for chaotic time series prediction, *IEEE Access*, **8** (2020), 159552–159565. https://doi.org/10.1109/ACCESS.2020.3020801
49. S. Bag, S. K. Kumar, M. K. Tiwari, An efficient recommendation generation using relevant Jaccard similarity, *Inf. Sci.*, **483** (2019), 53–64. https://doi.org/10.1016/j.ins.2019.01.023

50. A. A. Jalal, A. A. Jasim, A. A. Mahawish, A web content mining application for detecting relevant pages using Jaccard similarity, *Int. J. Electr. Comput. Eng.*, **12** (2022), 6461. https://doi.org/10.11591/ijece.v12i6.pp6461-6471
51. T. Ng, A. Lopez-Rodriguez, V. Balntas, K. Mikolajczyk, Reassessing the limitations of CNN methods for camera pose regression, *arXiv preprint*, 2021. https://doi.org/10.48550/arXiv.2108.07260
52. M. Waqas, U. W. Humphries, A critical review of RNN and LSTM variants in hydrological time series predictions, *MethodsX*, **13** (2024), 102946. https://doi.org/10.1016/j.mex.2024.102946
53. S. Manzhos, Q. G. Chen, W. Y. Lee, Y. Heejoo, M. Ihara, C. C. Chueh, Computational investigation of the potential and limitations of machine learning with neural network circuits based on synaptic transistors, *J. Phys. Chem. Lett.*, **15** (2024), 6974–6985. https://doi.org/10.1021/acs.jpclett.4c01413
54. N. E. Zamri, M. A. Mansor, M. S. M. Kasihmuddin, S. S. Sidik, A. Alway, N. A. Romli, et al., A modified reverse-based analysis logic mining model with Weighted Random 2 Satisfiability logic in Discrete Hopfield Neural Network and multi-objective training of Modified Niched Genetic Algorithm, *Expert Syst. Appl.*, **240** (2024), 122307. https://doi.org/10.1016/j.eswa.2023.122307
55. S. Z. M. Jamaludin, N. A. Romli, M. S. M. Kasihmuddin, A. Baharum, M. A. Mansor, M. F. Marsani, Novel logic mining incorporating log linear approach, *J. King Saud Univ.-Comput. Inf. Sci.*, **34** (2022), 9011–9027. https://doi.org/10.1016/j.jksuci.2022.08.026
56. G. Manoharam, M. S. M. Kasihmuddin, S. N. F. M. A. Antony, N. A. Romli, N. A. Rusdi, S. Abdeen, et al., Log-linear-based logic mining with multi-Discrete Hopfield Neural Network, *Mathematics*, **11** (2023), 2121. https://doi.org/10.3390/math11092121
57. N. A. Rusdi, M. S. M. Kasihmuddin, N. A. Romli, G. Manoharam, M. A. Mansor, Multi-unit Discrete Hopfield Neural Network for higher order supervised learning through logic mining: optimal performance design and attribute selection, *J. King Saud Univ.-Comput. Inf. Sci.*, **35** (2023), 101554. https://doi.org/10.1016/j.jksuci.2023.101554